NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A STUDY INTO THE EFFECTS OF KALMAN FILTERED NOISE IN ADVANCED GUIDANCE LAWS OF MISSILE NAVIGATION

by

Adam M. Osborn

March 2014

Thesis Advisor: Robert G. Hutchins
Second Reader: Xiaoping Yun

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2014	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE A STUDY INTO THE EFFECTS OF KALMAN FILTERED NOISE IN ADVANCED GUIDANCE LAWS OF MISSILE NAVIGATION	**5. FUNDING NUMBERS**
6. AUTHOR(S) Adam M. Osborn	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release;distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Advanced missile guidance laws may provide an air-to-air combat tactical advantage by increasing effective missile range. The current standard in missile guidance, proportional navigation (PN), is only optimal against a non-maneuvering target. Differential geometry (DG) guidance is optimized for a maneuvering target. Analysis of the DG guidance equation indicates noise degrades DG performance more than PN. This thesis evaluates the effect of Kalman filtered noise on PN and DG performance.

A simplified three degree of freedom (DOF) discrete time version of previous researchers' six DOF continuous time model is generated. Zero mean Gaussian white noise is inserted into simulated line-of-sight angle and range sensor measurements. Discrete time Kalman filters utilize these two noisy simulated sensor measurements to generate all guidance law inputs, including portions of the target state for DG. Simulations with Kalman filtered noise are conducted with both PN and DG guidance laws against maneuvering targets. Kinematic boundaries are used to evaluate a possible tactical advantage of DG over PN guidance in the presence of Kalman filtered noise.

14. SUBJECT TERMS Missile guidance, proportional navigation guidance law, differential geometry guidance law, Kalman filtering, Kalman filtered noise	**15. NUMBER OF PAGES** 173
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

A STUDY INTO THE EFFECTS OF KALMAN FILTERED NOISE IN ADVANCED GUIDANCE LAWS OF MISSILE NAVIGATION

Adam M. Osborn
Lieutenant, United States Navy
B.S.S.E., United States Naval Academy, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
March 2014**

Author: Adam M. Osborn

Approved by: Robert G. Hutchins
                    Thesis Advisor

                    Xiaoping Yun
                    Second Reader

                    R. Clark Robertson
                    Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Advanced missile guidance laws may provide an air-to-air combat tactical advantage by increasing effective missile range. The current standard in missile guidance, proportional navigation (PN), is only optimal against a non-maneuvering target. Differential geometry (DG) guidance is optimized for a maneuvering target. Analysis of the DG guidance equation indicates noise degrades DG performance more than PN. This thesis evaluates the effect of Kalman filtered noise on PN and DG performance.

A simplified three degree of freedom (DOF) discrete time version of previous researchers' six DOF continuous time model is generated. Zero mean Gaussian white noise is inserted into simulated line-of-sight angle and range sensor measurements. Discrete time Kalman filters utilize these two noisy simulated sensor measurements to generate all guidance law inputs, including portions of the target state for DG. Simulations with Kalman filtered noise are conducted with both PN and DG guidance laws against maneuvering targets. Kinematic boundaries are used to evaluate a possible tactical advantage of DG over PN guidance in the presence of Kalman filtered noise.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACEVAL   air combat evaluation

AIM   air intercept missile

AIMEVAL   air intercept missile evaluation

AMRAAM   advance medium range air-to-air missile

APN   augmented proportional navigation

AWG   air weapons guidance

DG   differential geometry

DOF   degrees of freedom

F   fighter aircraft

F/A   fighter / attack aircraft

FMS   foreign military sales

FOT&E   follow on test and evaluation

GAO   Government Accountability Office

GPS   Global Positioning System

JDRADM   joint dual role air dominance missile

JLENS   Joint Land Attack Cruise Missile Defense Elevated Netted Sensor System

LOS   line-of-sight

NED   north-east-down

NGM   next generation missile

PN   proportional navigation

RIO   radar intercept officer

SAM   surface to air missile

SARH   semi-active radar homing

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF SYMBOLS

$\| \ \|$	magnitude
$\wedge$	estimated value
Δ	discrete time step size
η	lead angle
η_M	missile lead angle
η_T	target lead angle
θ_L	line-of-sight angle
θ_{Lnz}	noisy line-of-sight angle
ρ	density
σ	standard deviation
σ_r	range sensor standard deviation
σ_{r_base}	range sensor baseline standard deviation
σ_{θ_L}	LOS angle sensor standard deviation
$\sigma_{\theta_L\_base}$	LOS angle sensor baseline standard deviation
a	acceleration
a_M	missile acceleration
a_{Ma}	azimuth missile acceleration
a_{Me}	elevation missile acceleration
a_{Mc}	commanded missile acceleration
a_{Mu}	deterministic missile acceleration
$a_{Mthrust}$	missile acceleration due to thrust
a_{Mdrag}	missile acceleration due to drag
a_T	target acceleration
a_{Tperp}	target acceleration perpendicular to the line-of-sight
a_{Tpara}	target acceleration parallel to the line-of-sight

xv

a_{turn}	target turn acceleration
e	elliptical planform
f_{noise}	noise multiplier
n_{rand}	pseudorandom zero mean Gaussian value
g	acceleration due to gravity
k	discrete time
κ	curvature
κ_M	curvature of missile maneuver
κ_T	curvature of target maneuver
m	missile mass
$q^2(k)$	process noise weighting factor
r	range
r_{nz}	noisy range
$u(k)$	deterministic input vector
v	velocity
v_c	closing velocity
v_M	missile velocity
v_T	target velocity
$\upsilon(k)$	process noise
$v(k)$	measurement residual
ω_{turn}	target angular turn rate
$w(k)$	measurement noise
$x(k)$	state
$x_M(k)$	missile x-axis position coordinate
$x_T(k)$	target x-axis position coordinate
$\hat{x}(k)$	predicted state estimate

$z(k)$	measurement
$\hat{z}(k)$	predicted measurement estimate
AR	wing aspect ratio
C_{dp}	parasitic drag coefficient
C_{di}	induced drag coefficient
C_{na}	azimuth induced drag coefficient
C_{ne}	elevation induced drag coefficient
E	expectation
F_{dp}	parasitic drag force
F_{di}	induced drag force
F_{ia}	azimuth induced drag force
F_{ie}	elevation induced drag force
$F(k)$	state transition matrix
$F_M(k)$	missile state transition matrix
F_T	target state transition matrix
$G(k)$	deterministic input gain matrix
$H(k)$	measurement extraction matrix
I	identity matrix
$M(k)$	missile state vector
$M_a(k)$	missile acceleration matrix
N	ratio of target and missile velocity magnitudes
N'	navigational constant
$P(k)$	state covariance
Q	dynamic pressure
$Q(k)$	process covariance
$R(k)$	measurement covariance

S_{REF}	missile cross-sectional area
$S(k)$	innovation covariance
$T(k)$	target state vector
$W(k)$	filter gain

EXECUTIVE SUMMARY

Air dominance is a crucial but rapidly deteriorating American wartime advantage. Evasive maneuvers improve with each generation of foreign fighter aircraft. The U.S. must expect our enemies to evolve. We must constantly strive to stay ahead of our adversary's technology if we are to remain the premier air power in the world. Improvements in air-to-air missile guidance can help maintain our aerial combat advantage.

Advanced missile guidance laws may provide an air-to-air combat tactical advantage by increasing effective missile range. The current standard in missile guidance, proportional navigation (PN), is only optimal against a non-maneuvering target. Differential geometry (DG) guidance is an advanced guidance law optimized for a maneuvering target. Analysis of the DG guidance equation indicates noise degrades DG performance more than PN.

This thesis evaluates the effect of Kalman filtered noise on PN and DG performance. A simplified three degree of freedom (DOF) discrete time version of previous researchers' six DOF continuous time model is generated. The model is based on open source characteristics of the AIM-120 advanced medium range air-to-air missile (AMRAAM). We assume onboard sensors provide actual position, velocity, and acceleration of the missile. Zero mean Gaussian white noise is inserted into simulated line-of-sight (LOS) angle θ_L and range r sensor measurements. Discrete time Kalman filters utilize these two noisy simulated sensor measurements and the missile's state vector to generate all guidance law inputs, including portions of the target state for DG. Simulations with Kalman filtered noise are conducted with both PN and DG guidance laws against maneuvering targets. The main objectives of this research are to:

- Determine the effect of Kalman filtered noise on the performance of PN and DG guidance laws.

- Determine if DG provides a tactical advantage over PN for guidance of the simulated AMRAAM in the presence of Kalman filtered noise.

The current standard in modern missile guidance is PN; it is simple, effective, and easy to implement [1]. It attempts to maintain the LOS angle constant as range closes. The PN commanded acceleration is [2]

$$a_{Mc} = \frac{N'v_c\dot{\theta}_L}{\cos(\eta_M)},$$

(1.1)

where v_c is the closing velocity, $\dot{\theta}_L$ is the LOS angle rate, and η_M is the angle between the LOS and the velocity vector of the missile also known as the missile lead angle. Since the missile state is assumed to be known, an estimate of η_M is also known based on the estimated LOS angle. The navigational constant N' is chosen to balance missile responsiveness with maximum range. For all simulations, $N'=5$. Commanded acceleration is applied in a direction perpendicular to the velocity vector of the missile as described in Figure 1 [2].

PURE PROPORTIONAL NAVIGATION
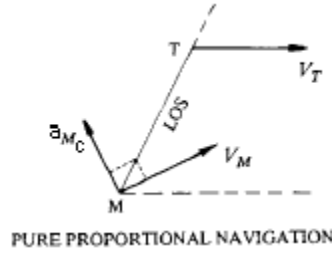
Figure 1.   Commanded acceleration application for proportional navigation (after [3]).

The DG guidance law uses classical differential geometry curve theory to navigate the missile to the target. It uses no predetermined target trajectory model. The magnitude of DG commanded acceleration is [4]

$$a_{Mc} = \|a_T\|\frac{\cos(\eta_T)}{\cos(\eta_M)} + \frac{N'v_c\dot{\theta}_L}{\cos(\eta_M)},$$

(1.2)

where η_T is the angle between the LOS and the velocity vector of the target and $\|a_T\|$ is the magnitude of the target's acceleration. Typical encounter geometry for DG is shown

in Figure 2. Similar to PN, DG commanded acceleration is applied perpendicular to the velocity vector of the missile.
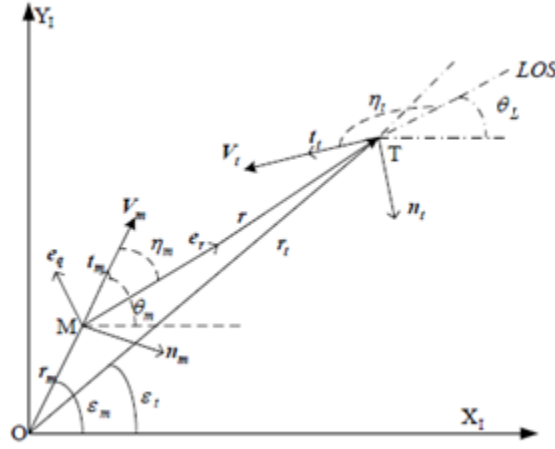
Figure 2.   Typical DG encounter geometry (after [5]).

The DG guidance law in (1.2) is remarkably similar to PN in (1.1). It applies PN guidance with an extra equation term that incorporates the curvature of the target maneuver. This extra term uses target parameters that must be extrapolated from noisy bearing and range measurements making DG quite susceptible to noise.

Each guidance law uses two Kalman filters to obtain the necessary parameters for guidance implementation; one filter estimates the LOS angle state, the other estimates the range state. These states consist of estimated time derivatives of LOS angle and range based on noisy measurements received from the simulated missile sensors. Since PN only requires the missile lead angle η_M, LOS angle rate $\dot{\theta}_L$ and closing speed v_c, the LOS angle and range states are

$$\hat{x}_{\theta_L}(k) = \begin{bmatrix} \hat{\theta}_L(k) \\ \hat{\dot{\theta}}_L(k) \end{bmatrix} \text{ and} \tag{1.3}$$

$$\hat{x}_r(k) = \begin{bmatrix} \hat{r}(k) \\ \hat{\dot{r}}(k) \end{bmatrix}. \tag{1.4}$$

Due to its increased complexity, DG filtering requires three dimensional theta and range states

$$\hat{x}_{\theta_L}(k) = \begin{bmatrix} \hat{\theta}_L(k) \\ \hat{\dot{\theta}}_L(k) \\ \hat{\ddot{\theta}}_L(k) \end{bmatrix} \text{ and} \qquad (1.5)$$

$$\hat{x}_r(k) = \begin{bmatrix} \hat{r}(k) \\ \hat{\dot{r}}(k) \\ \hat{\ddot{r}}(k) \end{bmatrix}. \qquad (1.6)$$

Estimates of target acceleration magnitude $\|\hat{a}_T\|$ and lead angle $\hat{\eta}_T$ are calculated using the third dimension time derivatives of LOS angle and range.

Unpredictable model inputs such as a target maneuver are modeled as zero mean white Gaussian process noise. Process covariance for two dimensional filters is [6]

$$Q(k) = q^2(k) \begin{bmatrix} \dfrac{\Delta^3}{3} & \dfrac{\Delta^2}{2} \\ \dfrac{\Delta^3}{2} & \Delta \end{bmatrix}, \qquad (1.7)$$

where Δ is the discrete step size in seconds. Process covariance for our three dimensional filters is [6]

$$Q(k) = q^2(k) \begin{bmatrix} \dfrac{\Delta^5}{20} & \dfrac{\Delta^4}{8} & \dfrac{\Delta^3}{6} \\ \dfrac{\Delta^4}{8} & \dfrac{\Delta^3}{3} & \dfrac{\Delta^2}{2} \\ \dfrac{\Delta^3}{6} & \dfrac{\Delta^2}{2} & \Delta \end{bmatrix}. \qquad (1.8)$$

Since the LOS angle filters estimate angular velocity and angular acceleration, the weighting factor $q^2(k)$ varies inversely with the square of range.

Noisy LOS angle and range measurements in the three DOF model are generated by multiplying a pseudorandom value drawn from a zero mean Gaussian distribution with the simulated sensor's standard deviation. A small standard deviation can have a significant effect on the maximum effective range of a missile. Measurement noise is interpreted by the missile guidance system as target acceleration. The guidance system responds to noise with slight guidance accelerations. These missile accelerations produce an induced drag that increases exponentially as missile speed approaches Mach 1. Small unnecessary guidance accelerations over the course of the entire missile flight can produce dramatic performance deterioration on the order of several kilometers. For the three DOF model, the range and LOS angle simulated sensor standard deviations are defined as 10 meters and 1 mrad as established by Pehr [10]. Noise with two to five times these standard deviations produces poor results for both PN and DG. This indicates the standard deviation values are reasonable in terms of measuring noise effects on performance. All Kalman filter models assume range and bearing measurements are unbiased and uncorrelated.

Numerous three DOF discrete time model simulations were run, generating kinematic boundaries we used to investigate the objectives of this thesis. Kinematic boundaries were used to measure guidance law performance. Much like an operating envelope, the kinematic boundary shows the maximum range from which a missile will destroy the target. The missile is launched from the envelope edge directly at the target. The target is at the origin pointing down the negative x-axis, which is the initial direction of target motion [7]. In the kinematic boundary, the aspect angle is measured from the positive x-axis to the LOS. An aspect angle of zero degrees would indicate a tail chase. An aspect angle of 180 degrees is a head-on geometry.

Instability in the aft quadrant appears in all kinematic boundaries in this thesis. The instability is likely due to transition from supersonic to subsonic speeds. This theory is supported by Pehr who experienced instability, which he eliminated by slightly smoothing the parasitic drag coefficient curve in the transonic region [4]. Kinematic boundaries can be reproduced for noiseless simulations but not for Kalman filtered noise simulations. Due to the pseudorandom nature of the Kalman filtered noise, the instability

in the aft quadrant is slightly different with each simulation. This can generate aft quadrant results in which maximum effective range with Kalman filtered noise is larger than noiseless range.

The kinematic boundaries for PN performance under noiseless and Kalman filtered noise conditions are shown in Figure 3. The kinematic boundaries are practically identical with the exception of instability in the aft quadrant. The range difference between the two is shown in Figure 4. Excluding the instability in the aft quadrant, slight constant noise degradation throughout the 360 degree view of about one kilometer can be seen.
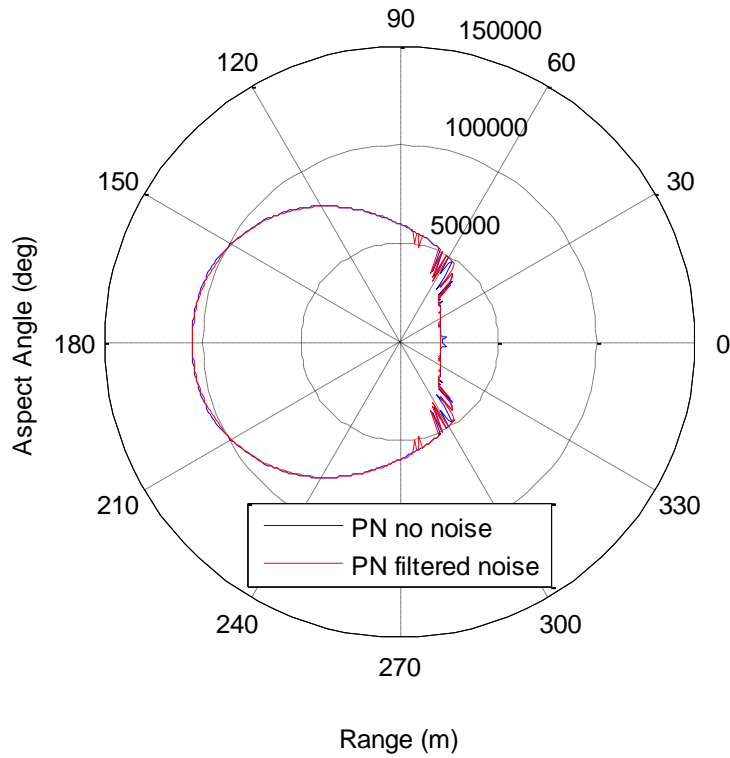
Figure 3. Kinematic boundaries for PN guidance under noiseless and Kalman filtered noise conditions.
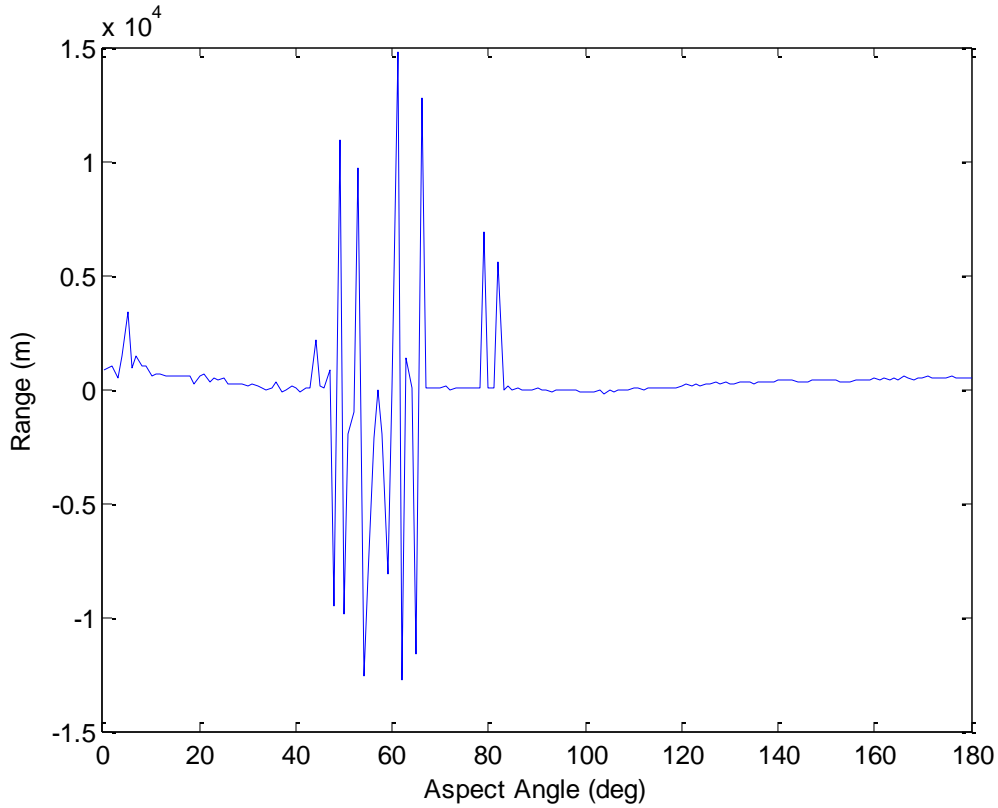
Figure 4.   Difference plot of PN maximum effective range under noiseless and
Kalman filtered noise conditions.

The DG guidance law has a more pronounced sensitivity to Kalman filtered noise than PN guidance. The kinematic boundaries for DG performance under noiseless and Kalman filtered noise conditions are shown in Figure 5. The Kalman filtered noise kinematic boundary is noticeable shorter in most aspect angles and experiences more pronounced instability in the aft quadrant. Instability in simulations with Kalman filtered noise may be compounded by the extreme noise sensitivity of DG target state estimates. The range difference between the kinematic boundaries is shown in Figure 6. Excluding the instability in the aft quadrant, noise degradation throughout the 360 degree view varies from one to six kilometers.
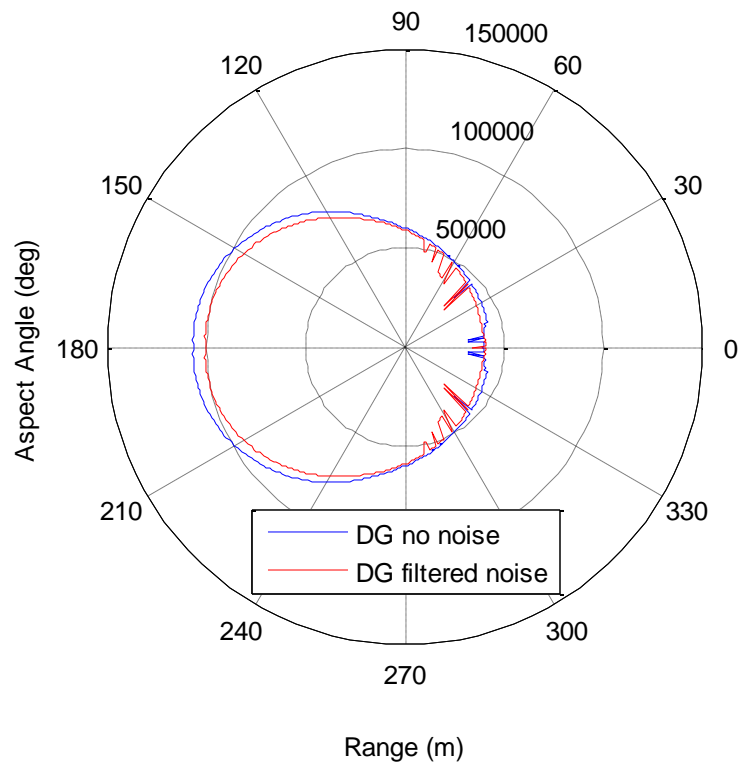
Figure 5.　　Kinematic boundaries for DG guidance under noiseless and Kalman filtered noise conditions.
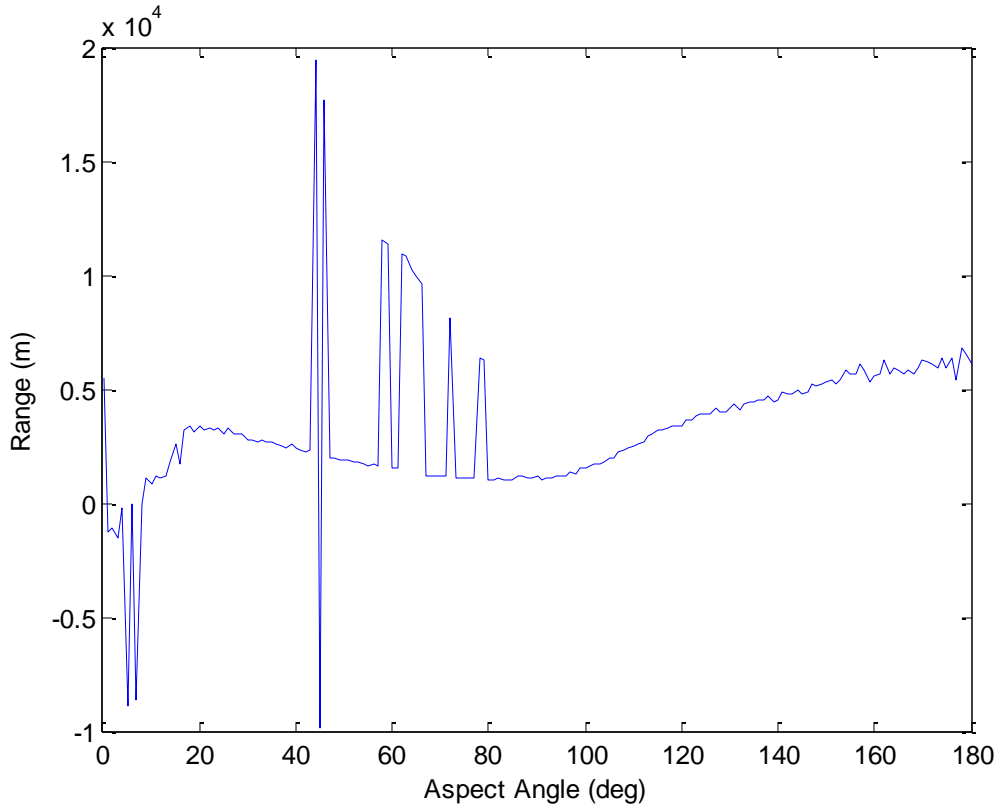
Figure 6.   Difference plot of DG maximum effective range under noiseless and Kalman filtered noise conditions.

DG does provide a tactical advantage over PN guidance. While DG guidance suffers slightly larger noise degradation in most aspect angles, it performs much better than PN in tail chase scenarios. The kinematic boundaries for DG and PN performance with Kalman filtered noise are shown in Figure 7. The range difference between the kinematic boundaries is shown in Figure 8. DG provides a maximum effective range that is extended between 17 to 20 kilometers beyond PN in the aft quadrant while sacrificing one to five kilometers in the remaining aspect angles. The effective range advantage in the aft quadrant outweighs the disadvantage in the other quadrants. These results serve as a point of departure for research of a hybrid combination of PN and DG guidance laws.
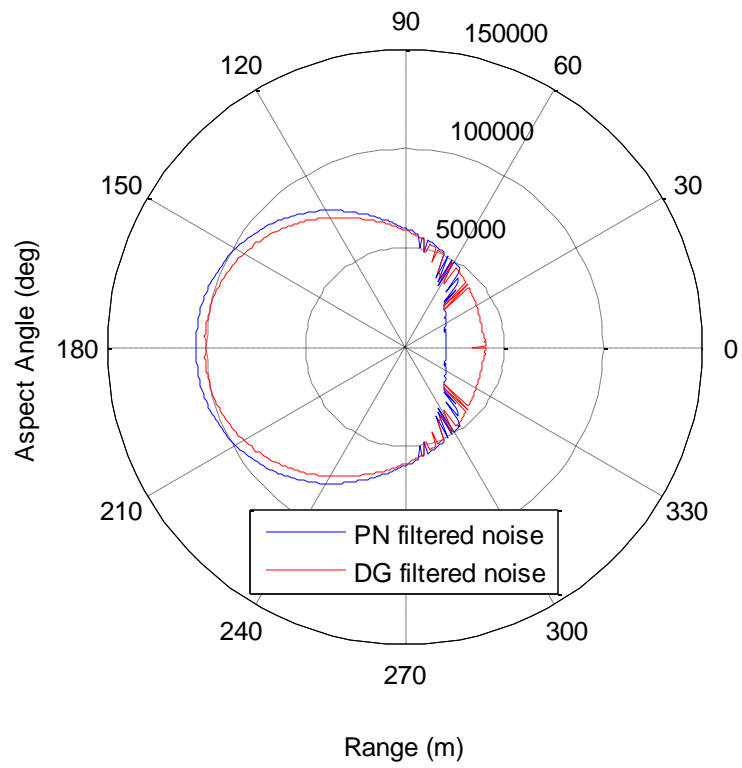
Figure 7.   Kinematic boundaries for PN and DG guidance with Kalman filtered noise.
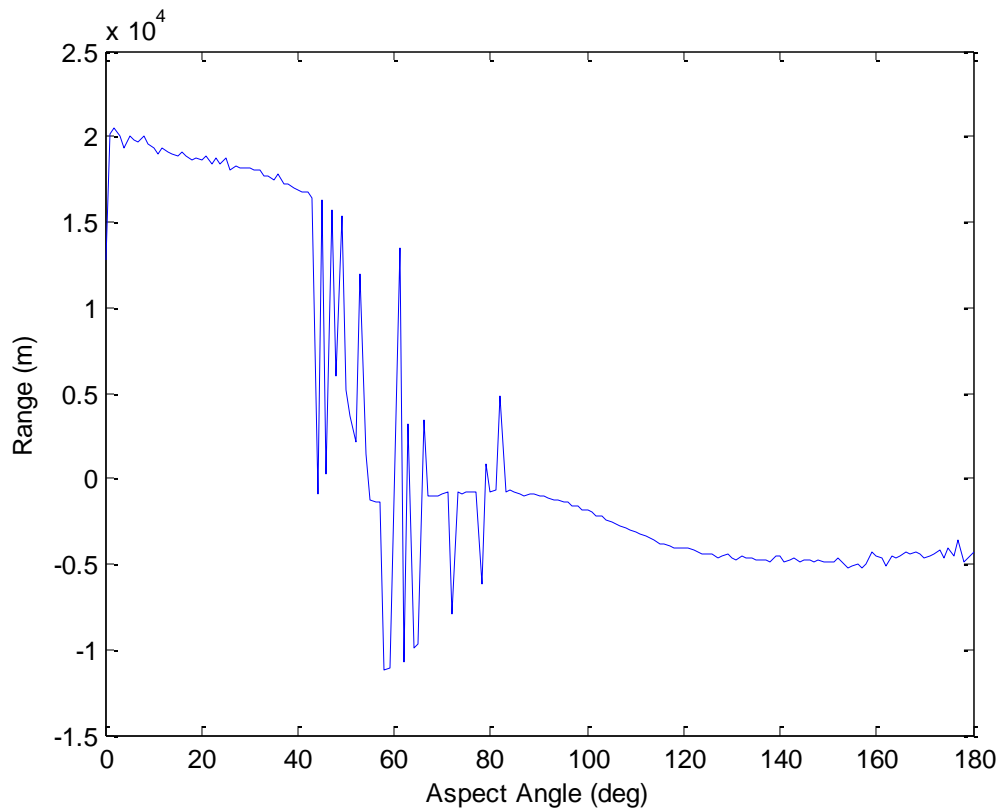
Figure 8.   Difference plot of PN and DG maximum effective range with Kalman
filtered noise.

The objectives of this research were met. The DG guidance law has a more pronounced sensitivity to Kalman filtered noise than PN guidance. DG guidance suffers slightly larger noise degradation in most aspect angles, but provides a distinct advantage over PN in tail chase scenarios. Based on these results we believe DG does provide a tactical advantage over PN guidance for the simulated AMRAAM.

LIST OF REFERENCES

[1]   P. Zarchan, Ed., *Tactical and Strategic Missile Guidance*, 3rd ed. Reston, VA: Amer. Inst. of Aeronautics and Astronautics, Inc., 1997.

[2]   R. G. Hutchins, "Navigation, Missile, and Avionics Systems," class notes [EC 4330], Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, January 2005.

[3]   U. S. Shukla and P. R. Mahapatra, "The proportional navigation dilemma—pure or true?" *IEEE Trans. on Aerospace and Electron. Syst.*, vol. 26, no. 2, pp. 382–392, Mar. 1990.

[4]   D. Pehr, "A study into advanced guidance laws using computational methods," M.S. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2011.

[5]   C. Li et al., "Application of 2D differential geometric guidance to tactical missile interception," presented at the IEEE Aerospace Conference, Big Sky, MT, 2006.

[6]   R. G. Hutchins, "Optimal Estimation, Kalman Filters and Target Tracking," class notes [EC 3310], Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, March 1997.

[7]   R. Broadston, "A method of increasing the kinematic boundary of air-to-air missiles using an optimal control approach," Elect. Eng. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2000.

ACKNOWLEDGMENTS

I would like to thank Professors Robert G. Hutchins and Xiaoping Yun for their guidance and assistance in this research. Thanks especially to Professor Hutchins for his invaluable patience and insight throughout this thesis.

I would also like to thank my wife and son, whose love and support made this thesis possible. Together it seems we can make anything happen.

THIS PAGE INTENTIONALLY LEFT BLANK

I.   INTRODUCTION

Air dominance is a crucial but rapidly deteriorating American wartime advantage. Evasive maneuvers improve with each generation of foreign fighter aircraft. The U.S. must expect our enemies to evolve. We must constantly strive to stay ahead of our adversary's technology if we are to remain the premier air power in the world. Improvements in air-to-air missile guidance can help maintain our aerial combat advantage.

Advanced missile guidance laws may provide an air-to-air combat tactical advantage by increasing effective missile range. The current standard in missile guidance, proportional navigation (PN), is only optimal against a non-maneuvering target. Differential geometry (DG) guidance is an advanced guidance law optimized for a maneuvering target. Previous research has shown that noise degrades DG performance more than PN. This thesis evaluates the effect of Kalman filtered noise on PN and DG performance.

A simplified three degree of freedom (DOF) discrete time version of previous researchers' six DOF continuous time model is generated. Zero mean Gaussian white noise is inserted into simulated line-of-sight (LOS) angle and range sensor measurements. Discrete time Kalman filters utilize these two noisy simulated sensor measurements to generate all guidance law inputs, including portions of the target state for DG. Simulations with Kalman filtered noise are conducted with both PN and DG guidance laws against maneuvering targets. Kinematic boundaries are used to evaluate a possible tactical advantage of DG over PN guidance in the presence of Kalman filtered noise.

This thesis is organized as follows. The remainder of Chapter I discusses the historical background and benefits of improving air-to-air missile guidance, previous work improving missile guidance, as well as the objectives of this thesis. Chapter II closely examines the PN and DG guidance laws. Chapter III provides an understanding of the Kalman filter, which can be used to improve guidance law performance. Chapter IV

describes the simulation environment created for this thesis. Chapter V contains simulation results and performance analysis. Chapter VI discusses our conclusions and recommendations for further research.

A. BACKGROUND

The AIM-7A Sparrow of the 1950s was one of the earliest air-to-air missiles in the arsenal of the United States [1]. This missile used a beam rider guidance system, which required the shooting aircraft to maintain radar lock on its target [1]. The missile would adjust its control surfaces in flight to "ride" a radar beam into the target. The early Sparrow's beam rider guidance was a simple design, giving it a distinct weight advantage. This form of guidance proved to be operationally useless as it was only successful against non-maneuvering targets, but radar-based guidance showed promise.

Sparrow's radar was a huge advantage over other guidance tools such as infrared. Its radar penetrated clouds and rain, allowing Sparrow to conduct terminal homing in all weather. Sparrow's radar-based guidance could also track low altitude targets due to its ability to pick out a target in spite of radar ground clutter. Later variants of Sparrow would capitalize on the advantages of radar.

The follow-on variant of Sparrow, AIM-7B, was ahead of its time [1]. AIM-7B used an active radar seeker, which sent radar pulses from the missile itself [1]. The onboard guidance would track the target without assistance from the shooting aircraft. This ability to "fire and forget" allows the shooter to engage multiple targets, which is crucial in modern air-to-air combat tactics. Unfortunately, implementation of this guidance was extremely heavy and complex prior to the advent of solid-state circuit technology. While the theory was sound, technological drawbacks kept this missile from being an operational reality [1]. A less advanced form of radar guidance would be needed for the next versions of Sparrow.

Later versions of the Sparrow, AIM-7D–E, were severely limited. The AIM-7E used semi-active radar homing (SARH) guidance. The parent aircraft illuminated the target while the missile only detected and interpreted the radar returns [2]. The SARH guidance system was more advanced than beam riding, but it proved to be a crucial

weakness of Sparrow. It required the pilot to fly directly at his target and not change course until the target was destroyed [2]. Such maneuver restrictions directly impact the effectiveness of the weapon.

By the 1970s, the standard air-to-air missile in the arsenal of the United States Air Force, AIM-7E Sparrow, was shown to be inadequate [3]. The air intercept missile evaluation / air combat evaluation (AIMEVAL/ACEVAL), conducted in the mid-1970s, proved the SARH guidance scheme put the pilot at significant risk [2]. The guidance system's requirement to fly toward the target meant the winner of an engagement would likely be the pilot who shot first [2]. The study showed pilots using Sparrow were at a severe disadvantage during multiple target engagements [2]. A pilot could only engage one target at a time and had little defense against a second target, producing a one-to-one kill ratio [2]. Such a low ratio was unacceptable during the Cold War era due to the quantitative advantage held by the Russian-led Warsaw Pact powers of Europe [2]. The Sparrow's form of guidance was not the only significant problem.

The Air Force F-16 Falcon was incompatible with Sparrow missiles [3]. Sparrow could be launched from the Navy's F-14 Tomcat and F/A-18 Hornet as well as the Air Force's F-15 Eagle as shown in Table 1. The Falcon was unable to carry Sparrow for two reasons. First, it lacked the avionics necessary to communicate with the Sparrow's guidance system [3]. Second, the 500 pound weight of the Sparrow precluded the missile from being mounted on the wingtip station of the Falcon [3]. The Falcon could carry Sparrow on the mounts under the body of the aircraft, but those spots were reserved for air-to-ground bombs during bombing missions [3]. The Air Force's intention was to have Falcon compose two thirds of its fleet by the 1990s. While Sparrow was only precluded from one aircraft, the effect was severe. Falcon had no radar-based air-to-air weapon.

Table 1.   Radar missile deployment by platform circa 1986 (from [3]).

	AIRCRAFT		
Service/ Country/ Aircraft Type	Missile System (Number per plane) Current	Planned	Current Aircraft Inventory
U.S. Air Force a/			
F-15	Sparrow (4)	AMRAAM (8)	730
F-16	None (Infrared Only)	AMRAAM (6)	910
F-4 b/	Sparrow (4)	None	1,220
U.S. Navy a/			
F-14	Sparrow (6) or Phoenix (6)	AMRAAM c/ (6) or Phoenix (6)	460
F/A-18	Sparrow (4)	AMRAAM d/ (6)	320
F-4 b/	Sparrow (4)	None	195
NATO Allies e/			

United Kingdom
  Tornado
  Sea Harrier
	Sky Flash f/
	(4)
	(0)
		AMRAAM
		(4)
		(2)
			
			g/
			30 g/

Germany			
F-4F	None (Infrared Only)	AMRAAM (4)	120

(Continued)

SOURCES: All data on U.S. ships and aircraft from U.S. Air Force and
Navy. UK aircraft data from British Defense Staff. German
aircraft data from AMRAAM Program Office. All ship data
from NATO Seasparrow Program Office.

a.   Lists maximum carriage; typical tactical carriage may be lower.
b.   U.S. Air Force and Navy F-4s are to be retired.

The Navy's AIM-54 Phoenix was not a viable substitute. The Phoenix was a long range missile with an advanced radar guidance system based heavily on the Tomcat AWG-9 radar. Using the Tomcat radar, the radar intercept officer (RIO) could simultaneously track 24 targets and allocate a priority target for each of the six Phoenix missiles onboard [4]. This missile was designed to shoot down Russian bombers at incredible ranges. It had the "fire and forget" capability of the AIM-7B, allowing a pilot

to engage multiple targets. Tomcat could also fire multiple missiles nearly simultaneously [3], [4]. These characteristics of active radar missiles like Phoenix were desperately needed on more platforms than the Tomcat.

The United States needed a new air-to-air missile. A joint service operational requirement was generated in 1978 for an advanced medium range air-to-air missile (AMRAAM) with active radar guidance. It was to be compatible with the Air Force's Eagle and Falcon as well as the Navy's Tomcat and Hornet [3]. This new missile would be a vast improvement over Sparrow. It was estimated the AMRAAM would have twice the combat capability at half the cost of Sparrow, resulting in a fourfold increase in cost efficiency [2]. The Air Force Chief of Staff had extremely high expectations. He estimated the Falcon's kill ratio would increase by a factor of six and Eagle's would double with AMRAAM [3]. Such vast improvement rarely comes easily.

Production and development of AMRAAM was quite a challenge. Both manufacturers, Hughes Aircraft and Raytheon, were to have test-fired 10 missiles each by December 1981; Raytheon tested five and Hughes only three [2]. Neither had developed a stable missile design [2]. By 1984, the cost per missile had more than doubled (in 1987 dollars) from $182,000 to $438,000, the development process was two years behind schedule, and there were regular calls from Congress to terminate the program [3]. The AMRAAM would likely have been scrapped without intense service support. The program was dramatically restructured in 1985, extending the timeline for production [2]. Secretary of Defense Caspar Weinberger was required by Congress to certify the program cost would not grow further and capabilities would not be reduced below current estimates [2]. This certification convinced Congress to continue funding in spite of a Government Accountability Office (GAO) report questioning Secretary Weinberger's certification and the viability of the program as a whole [2]. The AMRAAM proved to be a wise investment.

The AMRAAM developed into one of the most ubiquitous and highly capable missiles in the world. With the exception of the AIM-9 Sidewinder, AMRAAM has been evaluated more extensively than any other air-to-air missile [5]. During its final follow on test and evaluation (FOT&E) in the mid-1990s, 40 missiles were fired from 12 different

shot profiles [5]. The Director of Operational Test and Evaluation noted the missile reliability vastly exceeded user requirements [5]. In 1991, the GAO reported a pilot simultaneously engaged four targets with four AMRAAMs in the presence of electronic countermeasures, displaying both "fire and forget" and near simultaneous launch capabilities [6]. All four targets were considered kills [6]. The currently deployed AMRAAM, AIM-120C-7, is capable of intercepting anti-ship cruise missiles using the Joint Land Attack Cruise Missile Defense Elevated Netted Sensor System (JLENS) integrated radar system as shown in a recent test [7]. The AMRAAM was an international project with France, Germany and the United Kingdom nearly from its beginning [5]. It has been sold to 36 countries around the world via the United States' foreign military sales (FMS) program [7]. It is currently the standard air-to-air missile in the United States' arsenal.

The AMRAAM missile will be the premier medium range air-to-air missile for the foreseeable future. The newest AMRAAM, AIM-120D, has been upgraded significantly from its original form, incorporating Global Positioning System (GPS) navigation into the guidance system according to open source material [5], [7]. It is expected to be deployed in FY2014 [5]. The joint dual role air dominance missile (JDRADM), also referred to as the next generation missile (NGM), was slated to be the AMRAAM's replacement. The NGM program has been cancelled due to budget constraints [8]. Our focus should be on improving AMRAAM performance until the next generation of air-to-air missiles is developed and fielded. One way to improve performance is to upgrade the missile's guidance system.

Open source material indicates most modern active radar missile guidance systems utilize a form of the PN guidance law [9]. The ubiquity of PN guidance can be traced to its effectiveness and simplicity [9]. This guidance was first used in the United States' SAM-N-2 Lark prototype missile circa 1950 [9]. The guidance attempts to "lead" a target by pointing the missile ahead of the target and maintaining a constant LOS angle as range decreases. The LOS angle rate and closing velocity are all that is needed for PN guidance [9]. Infrared versions of PN guidance operate only on LOS angle rate and a rough guess of closing velocity [9]. The PN guidance law is capable of tracking a

maneuvering target and is the optimal guidance law for non-maneuvering targets. This law reacts to target maneuvers by rapidly reevaluating its course in an attempt to maintain a straight line to the collision point. The reactionary nature of PN tends to create superfluous guidance acceleration, wastes missile kinetic energy, and reduces the effective range of the missile. An advanced guidance law may reduce wasted energy and extend missile range.

The DG guidance law is optimized for maneuvering targets. It uses the target's magnitude of acceleration and lead angle to incorporate the curvature of a target maneuver into its commanded guidance. A lead angle is defined as the angle between the LOS and the velocity vector of either the missile or target. The DG guidance law has both advantages and disadvantages.

The DG guidance law theoretically wastes less kinetic energy reacting to target maneuvers, resulting in a longer effective missile range. It provides the missile with a more optimal path to the target, increasing the probability of a hit. An increase in effective range would be a distinct tactical advantage. Pilots would be able to engage enemies from greater distances, reducing the threat of the target returning fire. In a dogfight, the life of the pilot and a multi-million dollar aircraft hang in the balance. Any small tactical advantage can make a significant difference.

Accounting for target curvature has two distinct challenges. It requires more computation than PN and it is more sensitive to noise [10]. Filtering can reduce the effects of noise but further increases the computation requirements onboard the missile. Advances in modern micro-electronics have dramatically improved missile onboard processing power [11]. Based on the significant increase in computing capability, the DG guidance law may be operationally feasible. This method presents an opportunity for a tactical advantage over PN against maneuvering targets by extending a missile's effective range [10], [12].

B. RELATED WORK

Broadston conducted an in-depth comparison of optimal guidance laws in 2000 [11]. He created a six DOF Simulink™ model to compare the performance of five

optimal guidance laws, including PN. These laws were evaluated against maneuvering and non-maneuvering targets at a constant altitude of 6000 meters. When maneuvering, the target pulled a 6 g turn in the x-y plane beginning three seconds prior to missile impact [11]. He determined augmented proportional navigation (APN) provided a significant improvement over PN, but did not analyze DG [11]. The fact that APN accounts for target maneuver is significant to my research. It simply added a term to the PN guidance equation that applied a constant portion of target acceleration to the missile guidance acceleration [9]. He showed the practice of adding a term to account for target maneuver could increase guidance performance. The estimation of the target state used in this extra term makes it sensitive to noise.

Broadston conducted a brief noise study on a single approach angle; he inserted noise with a constant standard deviation into the actual range, range rate, LOS angle and LOS angle rate values [11]. Analyses of particular parameter noise effects were possible due to his "direct insertion" of noise, but this approach is unrealistic. A more accurate model of noise effects would simply provide the guidance system with noisy range and LOS angle measurements. The simulated missile in our model is equipped with range and LOS angle measurements only. Range rate and LOS angle rate are extrapolated from these inputs.

Broadston introduced the kinematic boundary to quantify the performance differences between guidance laws. The kinematic boundary shows a 360 degree view of the maximum range from which a missile will reach the kill radius of five meters and cause "substantial, if not fatal damage" to the target aircraft [11]. Similar to an operating envelope, the kinematic boundary is an intuitive tool for analyzing missile performance [11]. The kinematic boundary was used by Pehr in continuation of Broadston's work [10]. It will be discussed in detail in Chapter IV.

Pehr conducted an analysis of APN, PN and DG performance and their response to unfiltered noise against maneuvering and non-maneuvering targets in 2011 [10]. He continued use of Broadston's six DOF model and the "direct insertion" of white noise into signal parameters including target position, velocity and acceleration [10]. He established a noise testing range 10 percent below the noiseless maximum range to

measure sustainable noise. Pehr determined the maximum sustainable noise to be that in which the missile scored a hit in at least 70 out of 100 simulations at the noise testing range [10].

Pehr's results showed DG to be superior overall [10]. With no noise applied, DG showed significant improvement over PN and APN in chasing maneuvering targets [10]. When unfiltered noise was applied, DG maintained a performance advantage over APN and PN when engaging maneuvering targets [10].

Comparison of DG, PN, and APN versus non-maneuvering targets in the presence of unfiltered noise revealed an interesting result. Both APN and DG experience significant degradation in performance with noise applied [10]. The PN law showed much less degradation in the presence of noise. This result supported the inference by Broadston that PN is optimal under a certain set of conditions, namely a non-maneuvering target [11]. Pehr concluded the more extensive unfiltered noise degradation of APN and DG performance is intuitive since these laws are more complex [10].

The implementation of any guidance law is subject to the noise encountered in onboard sensors providing information about the target. It is very likely modern missiles incorporate filtering to combat the effects of noise. The DG guidance law has more sensitivity to noise since it is estimating more detailed information about the target trajectory. Both the DG and PN kinematic boundary could benefit from the filtering of this sensor noise. A Kalman filter is a common and effective discrete time linear state estimator that can mitigate the effects of noise. A logical question following Pehr's work is how Kalman filtered noise will affect the performance of DG and PN.

C. OBJECTIVES

This thesis builds on previous research of the PN and DG guidance laws. The objectives of this thesis are fourfold:

- Determine the effect of Kalman filtered noise on the performance of PN and DG guidance laws.

- Determine if DG provides a tactical advantage over PN for guidance of the simulated AMRAAM in the presence of Kalman filtered noise.

9

- Determine what level of Kalman filtered noise these guidance laws can withstand.
- Explore the effect of discrete time measurement sample rate on guidance law performance.

Our objectives are achieved by creating a discrete time three DOF MATLAB® model based on open source characteristics of the AIM-120 AMRAAM. The discrete time model allows us to control the sample rate and use discrete time measurement noise. We assume the actual missile state is known due to accelerometers and a GPS transceiver onboard the simulated missile. Due to this assumption, the actual missile state vector is made available for all guidance law calculations. We validate our three DOF model by comparing the maximum missile range without noise for a stationary target with that of Broadston and Pehr.

For simulations with noise, we implement a zero mean Gaussian white noise in range and LOS angle measurements. We use discrete time Kalman filters to generate all required guidance law inputs, including the target's magnitude of acceleration and lead angle, from these two noisy sensor inputs. The kinematic boundary in simulations with Kalman filtered noise represents the maximum range from which the missile reaches the kill radius in at least 14 of 20 simulations. The number of simulations was chosen to balance the time required to generate a kinematic boundary and the appearance of statistical anomalies while still maintaining the 70 percent efficiency standard set by Pehr [10]. While our noise insertion method is more realistic than Broadston and Pehr's "direct insertion" method, it does not permit a direct comparison with these earlier results.

We compare kinematic boundaries under noiseless and Kalman filtered noise conditions to determine the effect of Kalman filtered noise on guidance law performance. We use these effects to evaluate the tactical advantage of DG over PN at the baseline noise level. The noise sensitivity associated with PN and DG is also evaluated.

We determine the maximum sustainable filtered noise level in a fashion similar to Pehr. Simulations with increasing noise are run at a noise testing range 10 percent below the maximum noiseless range. The maximum sustainable Kalman filtered noise level is that in which the missile reaches the kill radius in at least 14 of 20 simulations.

10

We generate kinematic boundaries with Kalman filtered noise using a sample rate of 5, 10 (baseline), and 20 milliseconds. We evaluate the effect of sample rate on PN and DG guidance law performance by comparing these kinematic boundaries.

In this chapter, we briefly discussed the current state of missile guidance. Previous generations of missile guidance have shown the superiority of onboard active radar guidance. The AMRAAM has been defined as the standard air-to-air missile in the U.S. arsenal for the foreseeable future. The DG advanced missile guidance has been identified as a possible way to improve the AMRAAM. Previous work in this area by Broadston and Pehr has shown DG provides an extended range for AMRAAM against maneuvering targets with "direct insertion" noise. Our main objectives in this thesis are to investigate the effects of Kalman filtered noise on the performance of PN and DG guidance laws and determine if DG can provide a tactical advantage over PN for guidance of the simulated AMRAAM in the presence of Kalman filtered noise. To achieve these objectives, we must better understand the PN and DG guidance laws discussed in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK.

II. GUIDANCE LAWS

In this chapter, we discuss the mechanics of the PN and DG guidance laws. The PN guidance law is optimized for a non-maneuvering target, meaning it assumes the target is traveling in a straight line. This is seldom the case in a real-world scenario. The DG guidance law does not use a fixed model of target maneuver. It uses classical differential geometry curve theory to incorporate target maneuver curvature into the missile's guidance [10].

The following is the notation for this section. A dot over a symbol indicates a time derivative. Angles will be delineated by θ_L for the LOS angle or η for a lead angle. Parameters specific to the missile or target are given a subscript. For example, the missile lead angle is η_M. Navigational constant N' should not be confused with N, which is the ratio of target and missile velocity magnitudes. Vector lengths are annotated with double bars such as target speed $\|v_T\|$. Commanded accelerations will be denoted by a_{Mc}.

A. PROPORTIONAL NAVIGATION

The proportional navigation (PN) guidance law is the current standard in modern missile guidance; it is simple, effective, and easy to implement [9]. This guidance attempts to maintain the LOS angle θ_L constant as range r closes. A collision is assured if the bearing to an object does not change as range decreases. The LOS angle is measured from a fixed coordinate axis, in this case the x-axis, to the LOS. The only information needed to implement PN guidance is LOS angle rate $\dot{\theta}_L$ and closing speed

$$v_c = -\dot{r}.$$ (2.1)

The PN commanded acceleration is [9]

$$a_{Mc} = N'v_c\dot{\theta}_L.$$ (2.2)

The navigational constant N' is chosen to balance missile responsiveness with maximum range. For all simulations $N' = 5$ as was done by Pehr [10].

The direction of commanded acceleration depends on the type of PN guidance being implemented. In the strictest sense of application, commanded acceleration should be applied perpendicular to the instantaneous LOS [9]. This is known as "true" PN [13]. As a matter of practical implementation of missile guidance, the commanded acceleration is applied in a direction perpendicular to the velocity vector of the missile [14]. This is known as "pure" PN and is the method implemented in this thesis [13]. These geometries are described in Figure 1.

TRUE PROPORTIONAL NAVIGATION   PURE PROPORTIONAL NAVIGATION
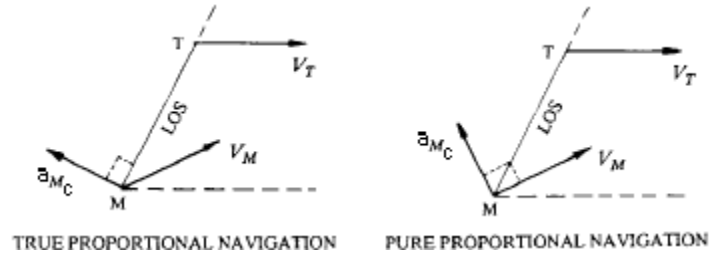
Figure 1.   Commanded acceleration application for "true" and "pure" proportional navigation (after [13]).

It is apparent in Figure 1 that the effect on the missile body is changed with the direction of acceleration. The commanded "pure" PN acceleration must be increased such that its projection onto the vector perpendicular to the LOS is of the same magnitude as the "true" PN acceleration. This modification is achieved using the missile's lead angle η_M. The modified commanded acceleration is [14]

$$a_{Mc} = \frac{N'v_c\dot{\theta}_L}{\cos(\eta_M)}.$$

(2.3)

The missile lead angle is expected to be on the order of a few degrees. Application of "pure" PN slightly increases the portion of acceleration in the LOS. This effect on the performance of the guidance is negligible due to the small size of the missile lead angle.

B.   DIFFERENTIAL GEOMETRY

The differential geometry (DG) guidance law utilizes classical differential geometry curve theory to navigate the missile to the target. This optimizes DG for a

maneuvering target by analyzing target trajectory curvature vice using a predetermined target trajectory model. Typical encounter geometry for DG is shown in Figure 2.

Figure 2.   Typical DG encounter geometry (after [12]).

The curvature of the missile κ_M is [12]
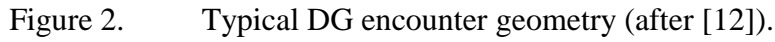
$$\kappa_M = N^2 \kappa_T \frac{\cos(\eta_T)}{\cos(\eta_M)} + \frac{N' v_c \dot{\theta}_L}{v_M^2 \cos(\eta_M)},$$ (2.4)

where κ_T is the target curvature, η_T is the target lead angle, and N is [12]

$$N = \frac{\|v_T\|}{\|v_M\|}.$$ (2.5)

A general equation for the curvature of a circular motion is

$$\kappa = \frac{\|a\|}{\|v\|^2},$$ (2.6)

where a is acceleration. Curvature of a circular motion is defined as a positive value and does not differentiate between a curve to the left and right. A unit vector perpendicular to the LOS extending to the left is used as the direction of commanded acceleration in this thesis. This vector allows us to distinguish left and right and alters the curvature equation for the missile to

15

$$\kappa_M = \frac{a_{M_c}}{v_M{}^2}. \tag{2.7}$$

Substituting (2.5), (2.6), and (2.7) into (2.4), yields [10]

$$\left(\frac{a_{Mc}}{v_M^2}\right) = \left(\frac{\|v_T\|}{\|v_M\|}\right)^2 \left(\frac{\|a_T\|}{\|v_T\|^2}\right) \frac{\cos(\eta_T)}{\cos(\eta_M)} + \frac{N'v_c\dot{\theta}_L}{v_M^2\cos(\eta_M)}. \tag{2.8}$$

Simplifying (2.8), produces commanded acceleration [10]

$$a_{Mc} = \|a_T\| \frac{\cos(\eta_T)}{\cos(\eta_M)} + \frac{N'v_c\dot{\theta}_L}{\cos(\eta_M)}. \tag{2.9}$$

In Chapter II, we discussed construction of the PN and DG guidance laws. The PN guidance assumes a non-maneuvering target model and strives to maintain a constant LOS angle. The DG guidance uses differential geometry curve theory, which avoids a specific target maneuver model. The DG guidance law in (2.9) is remarkably similar to PN in (2.3). It applies "pure" PN guidance with an extra equation term that incorporates the curvature of the target maneuver using the magnitude of the target's acceleration $\|a_T\|$ and lead angle η_T. These target parameters must be extrapolated from bearing and range measurements making DG quite susceptible to noise. In Chapter III, we discuss the Kalman filter, which we will use to mitigate the effects of noise in both PN and DG guidance.

III.   KALMAN FILTERING

The Kalman filtering algorithm is a highly effective linear state estimator. Known as the workhorse of estimation, the discrete time Kalman filter uses a recursive minimum mean square error process to correct a predicted state estimate after each measurement [15]. Put simply, a Kalman filter can estimate the position, velocity, and acceleration of a parameter given only noisy estimates of position.

In this chapter, we describe the Kalman filtering process using the following notation. A particular discrete time step is referred to as the time and is annotated with k. A vector at time k will be referred to with a lower case letter such as $x(k)$. Matrices at time k are annotated with the capital letters such as $P(k)$. A hat above a symbol signifies an estimate. For example $\hat{x}(k+1|k)$ will be read, "the estimated state at time $k+1$ given the information from all previous time steps up to and including time k." Magnitudes are annotated with double bars such as $\|a\|$. Matrix inversion and transpose will be shown as a superscript, $S(k)^{-1}$ and $S(k)'$ respectively.

The Kalman filter algorithm can be broken into prediction and correction phases [14]. The prediction phase occurs at the end of the current time k. It calculates a predicted state estimate and covariance for the next time $k+1$ using the current time corrected state estimate and any known deterministic inputs. The measurement is taken between the prediction and correction phases and marks the beginning of time $k+1$ [14]. The correction phase then generates a corrected state estimate and covariance for time $k+1$. The flow of the algorithm is shown in Figure 3 [15]. Before we discuss prediction and correction phases further, we must describe the dynamic plant used to model a linear dynamic system.
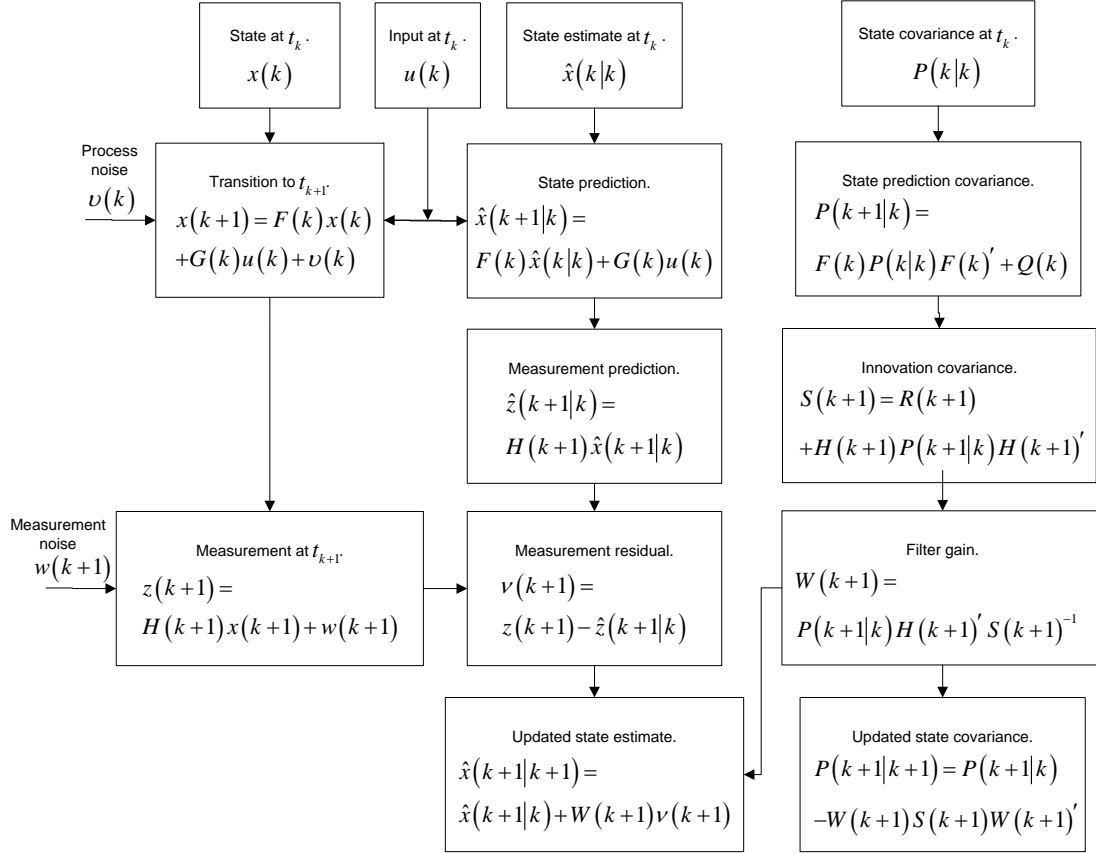
Figure 3.   One cycle in the state estimation of a linear system (from [15]).

A.   DYNAMIC PLANT

The dynamic plant is the mathematical model of a system. Process noise is used to model unpredictable state variation. Measurement noise is used to model unpredictable variation in measurements. Each of these terms has an associated covariance that describes its accuracy. To understand the dynamic plant, we must first examine the state equation.

1.   State Equation

The state equation describes a linear dynamic system using a vector difference equation. The state vector is the smallest vector that summarizes a deterministic system in full [15]. The updated state vector is [15]

$$x(k+1) = F(k)x(k) + G(k)u(k) + \upsilon(k), \tag{3.1}$$

where $k = 0, 1, \ldots$. The state transition matrix $F(k)$ determines the change in state due only to the effect of time from k to $k+1$. The deterministic input vector $u(k)$ accounts for known deterministic state inputs. The deterministic input gain matrix $G(k)$ is an identity matrix that can be used to weight $u(k)$. Process noise $\upsilon(k)$ is a zero mean white Gaussian noise. The state equation uses process noise to account for unpredictable state variation. The output of the state equation is the state vector $x(k+1)$, from which the measurement $z(k+1)$ is derived. A typical three dimensional state vector is

$$x(k+1) = \begin{bmatrix} z(k+1) \\ \dot{z}(k+1) \\ \ddot{z}(k+1) \end{bmatrix}. \tag{3.2}$$

2.   Measurement

The measurement $z(k+1)$ encompasses state translation and transients during the time elapsed between steps as well as measurement noise. It occurs between the prediction and correction phases at the beginning of time $k+1$. The measurement at time $k+1$ is [15]

$$z(k+1) = H(k+1)x(k+1) + w(k+1). \tag{3.3}$$

The measurement extraction matrix $H(k+1)$ draws the measurement from the state vector for time $k+1$. Measurement noise $w(k+1)$ is a zero mean white Gaussian noise that describes unpredictable variation in the measurement at time $k+1$ [15].

3.   Plant Covariance

Covariance is a measure of a variable's accuracy at the current time. Measurement covariance $R(k)$ and process covariance $Q(k)$ are assumed to be

mutually independent [15]. Measurement covariance $R(k)$ is the covariance of measurement noise $w(k)$ and [15]

$$R(k) = E\left[w(k)w(k)' \right],$$
(3.4)

where E is the expectation. Measurement covariance is based on the standard deviation σ of the measurement. Hence, $R(k)$, typically a known constant in tracking applications, is [14]

$$R(k) = \left[\sigma^2 \right].$$
(3.5)

Process covariance $Q(k)$, the covariance of zero mean white Gaussian process noise, is [15]

$$Q(k) = E\left[\upsilon(k)\upsilon(k)' \right].$$
(3.6)

It is based on the largest unpredictable state variation expected in the system being modeled. A large process covariance indicates the filter should expect large changes in measurements. When process covariance is large, the filter discounts prior estimates and measurements [14]. This allows the filter to maintain track in the event of large unpredictable state variation but inserts significant uncertainty when the state is constant [14]. A small process covariance discounts the current measurement, which allows for more precise estimation of a constant state, but limits the filter's ability to track through large unpredictable state variations.

The state covariance $P(k)$ reflects the accuracy of the filter's state estimate at time k. Assuming the state being modeled is known at time $k = 0$, state covariance is initialized with the measurement covariance. Over several time steps, the filter mitigates the effects of noise and creates more accurate estimates with lower covariance [14]. As more measurements are taken, state covariance is reduced until it reaches a minimum

steady state value based on the measurement covariance $R(k)$ and process covariance $Q(k)$.

B.   PREDICTION PHASE

The prediction phase uses the corrected state estimate and covariance, as well as any known inputs from the current time k, to calculate the predicted state estimate and covariance for time $k+1$ [16]. This is done prior to collecting the actual noisy measurement at time $k+1$ [16]. The products of the prediction phase are: the predicted state estimate, predicted state covariance, and predicted measurement estimate. All of these are utilized in the correction phase to update the filter's knowledge of the state vector's past and improve its ability to predict the state's future.

1.   Predicted State Estimate

The state prediction equation generates a predicted state estimate at time $k+1$ using only measurements up to time k. The state prediction equation is [15]

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k). \tag{3.7}$$

The predicted state estimate accounts for the state translation due to time and the deterministic input. It is the filter's best guess of the state vector at time $k+1$.

2.   Predicted State Estimate Covariance

The predicted state estimate covariance $P(k+1|k)$ estimates the accuracy of the predicted state estimate based on the corrected state covariance and process covariance of the previous time step k. It is given by [15]

$$P(k+1|k) = F(k)P(k|k)F(k)' + Q(k). \tag{3.8}$$

Predicted state estimate covariance $P(k+1|k)$ accounts for the possibility of an unpredictable state variation from time k to $k+1$ with process noise covariance $Q(k)$.

21

The addition of process noise covariance causes the predicted state estimate covariance to increase.

3. Predicted Measurement Estimate

The predicted measurement estimate is what the filter expects the measurement to be at time $k+1$ based on its predicted state estimate, and is [15]

$$\hat{z}\left(k+1|k\right)=H\left(k+1\right)\hat{x}\left(k+1|k\right). \tag{3.9}$$

C. CORRECTION PHASE

The correction phase completes the Kalman filter cycle by updating the predicted state estimate and covariance to generate the corrected state estimate and covariance. In addition to the results of the prediction phase and measurement, measurement residual and filter gain are utilized in the correction phase. The update to the predicted state estimate will be proportional to the measurement residual and the filter gain. We must further explain these two factors as they dictate how quickly the corrected state estimate will adjust to unpredictable state variations. Only then can we discuss development of the corrected state estimate and corrected state estimate covariance.

1. Measurement Residual

Measurement residual $\nu\left(k+1\right)$ is the difference between the measurement and the predicted measurement estimate, i.e., [15]

$$\nu\left(k+1\right)=z\left(k+1\right)-\hat{z}\left(k+1|k\right). \tag{3.10}$$

Large measurement residuals indicate the measurement and predicted measurement estimate are far apart.

2. Filter Gain

Filter gain controls the influence of previous estimates and the current measurement when updating the predicted state estimate and covariance. A small gain

increases the influence of previous measurements while a large gain emphasizes the current measurement. Filter gain $W(k)$ is [15]

$$W(k+1) = P(k+1|k) H(k+1)' S(k+1)^{-1}. \tag{3.11}$$

The innovation covariance $S(k+1)$, the covariance of the measurement residual $v(k+1|k)$, is [15]

$$S(k+1) = R(k+1) + H(k+1) P(k+1|k) H(k+1)'. \tag{3.12}$$

3.   Corrected State Estimate

The corrected state estimate, which adjusts the predicted state estimate by a magnitude proportional to the measurement residual and the filter gain, is

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1) v(k+1). \tag{3.13}$$

Provided the dynamic plant accurately models the actual state and measurement processes, the corrected state estimate will tend toward the actual system state. A non-zero process noise prevents the corrected state estimate from reaching the actual state but over many time steps the corrected state estimate has less error than the measurements alone. In this manner, the effects of measurement noise are mitigated.

4.   Corrected State Estimate Covariance

Over the course of several discrete time steps, corrected state covariance reduces to a steady state value independent of unpredictable state variations. In (3.8) predicted state estimate covariance $P(k+1|k)$ expands by adding process covariance to account for unpredictable state variations The corrected state covariance $P(k+1|k+1)$ contracts due to the new measurement. Over several time steps, the contraction in the correction phase gets smaller. Assuming the state and measurement covariance structure remains constant, the expansion in prediction and contraction in correction eventually reach parity, leaving a steady state corrected value. The corrected state covariance is [15]

$$P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W(k+1)'. \qquad (3.14)$$

The corrected state estimate covariance in (3.14) is more sensitive to rounding errors and can be unstable due to generating negative eigenvalues [15]. The Joseph form of the corrected state covariance shown in (3.15) corrects these issues [15]. While more computationally expensive, the improvement in mathematical behavior is quite desirable [15], [16]. The Joseph form of corrected state estimate covariance is [15]

$$P(k+1|k+1) = \left[I - W(k+1)H(k+1) \right] P(k+1|k) \left[I - W(k+1)H(k+1) \right]' \cdots \\ + W(k+1)R(k+1)W(k+1)'. \qquad (3.15)$$

In Chapter III, we have discussed the Kalman filter algorithm. The dynamic plant models a system. The prediction phase generates a predicted state estimate and covariance at time k for time $k+1$. The measurement marks the beginning of time $k+1$. The correction phase completes the Kalman filter's cycle by generating a corrected state estimate and covariance for time $k+1$ that reduces the mean square of the measurement residual of previous time steps. The general Kalman filter algorithm must be tailored to each application. In Chapter IV, we examine simulation methodology. We will describe the three DOF model in detail as well as the Kalman filter modifications necessary for this thesis.

IV. THREE DOF SIMULATION METHODOLOGY

The six DOF continuous time model derived by Broadston is simplified to a discrete time three DOF model. The depths of Broadston's six DOF model in describing moments of inertia acting on a missile body in flight are overly complicated for this thesis. The point mass three DOF model adequately represents a missile for our purposes. Discrete time permits a Kalman filter at a specified measurement sample rate.

In this chapter, we will study the simulation methods used in the three DOF discrete time model. We begin by examining the translation of the simulation into a kinematic boundary. The structure of missile and target motion and the effects of drag are carefully implemented to ensure the same in-flight characteristics as the six DOF model. Our method of discrete time noise injection is devised. We end this chapter by tailoring the Kalman filter and guidance laws to suit our application.

Notation used in this chapter is as follows. The discrete time step size Δ is defined as the time elapsed between discrete time steps k and $k+1$. A dot above a symbol indicates a time derivative; two dots, a second derivative. A hat above a symbol signifies an estimate. Magnitudes are annotated with double bars $\parallel \parallel$. The capital letters M and T will refer to the missile and target respectively. For example, $M(k)$ is the missile's state vector while $x_M(k)$ is the missile's x-axis position coordinate at time k.

A. KINEMATIC BOUNDARY CONSTRUCTION

In this section we will establish an understanding of the kinematic boundary and its construction from several simulations. Broadston introduced the kinematic boundary to quantify the performance differences between guidance laws [11]. Pehr also used this measure of performance, but slightly modified the orientation [10]. The orientation of the simulation and kinematic boundary in this thesis are preserved from Pehr's work.

Kinematic boundary geometry has the missile launched from the envelope edge directly at the target. The target is at the origin pointing down the negative x-axis, which is the initial direction of target motion [11]. The aspect angle is measured from the

positive x-axis to the LOS. An aspect angle of zero degrees would indicate a tail chase. An aspect angle of 180 degrees is a head-on geometry. Much like an operating envelope, the kinematic boundary shows the maximum range from which a missile will reach the kill radius. The kill radius is defined as a five meter sphere around the center of the target, inside which a missile explosion would cause "substantial, if not fatal damage" to the target aircraft [11]. An example of a kinematic boundary is shown in Figure 4.
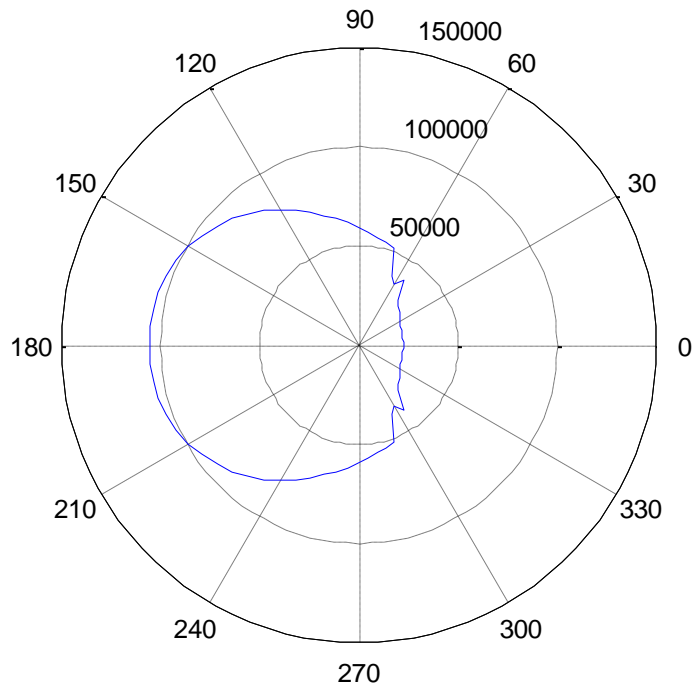
Figure 4.   Example of a kinematic boundary.

The geometry of the three DOF simulation is translated to create the kinematic boundary. It allows the missile and target to begin on the x-axis in all simulations while maintaining the desired aspect angle. At the beginning of each three DOF simulation, the missile is initialized at the origin, pointing at the target. The target is initialized on the positive x-axis, separated from the missile by the range being tested. The initial heading of the target, measured from the x-axis, is set to the desired aspect angle.

26

The kinematic boundary is a locus of points constructed from many three DOF simulations. Similar to Broadston and Pehr, the three DOF simulation concludes if any of the following three conditions are met:

- The missile velocity decreased below target velocity, indicating the missile no longer has the kinetic energy to chase down the target.

- The range to target begins to open, indicating the missile has missed the target.

- The missile comes within the kill radius (five meters) of the target, indicating a hit.

Simulations are run at the specified aspect angle with increasing ranges until the missile can no longer reach the kill radius. The maximum range of successful interception of the target is presented as a point on the kinematic boundary at the tested aspect angle. Beginning with zero degrees, the aspect angle is incremented and the process is repeated, ultimately creating a 180 degree arc. The 180 degree arc of the kinematic boundary is mirrored to produce a 360 degree view.

The results displayed in the kinematic boundary are greatly affected by the flight characteristics of the missile and target. An accurate missile model is the first step in creating meaningful kinematic boundaries.

B.   MISSILE MODEL

In this section, we will describe our model for missile motion and thrust. The missile model must account for the effects of gravity, parasitic and induced drag, engine thrust, and commanded guidance in flight. We must also prevent these forces from exceeding the capability of a typical missile frame.

The missile is modeled as a point mass in the North-East-Down or NED coordinate system. The NED coordinate system dictates an x-axis pointing north, y-axis to the East and z-axis down toward the center of the Earth. This coordinate system assumes the surface of the Earth is approximated by the flat tangent plane, limiting the effective simulation range to less than 200 kilometers. The Earth pointing z-axis allows for a constant gravitational pull along this axis. Altitude above the surface of the Earth using NED coordinates is specified as a negative number.

1. Missile Motion

Missile motion is described using a state vector with position and velocity components along the x, y, and z axes. The velocity vector is considered to be directly through the imaginary "frame" or central axis of the missile body due to the small angles of attack the missile will experience. The missile state vector $M(k)$, which describes the position and velocity of the missile at any given discrete time step k, is

$$M(k) = \begin{bmatrix} x_M(k) \\ \dot{x}_M(k) \\ y_M(k) \\ \dot{y}_M(k) \\ z_M(k) \\ \dot{z}_M(k) \end{bmatrix}. \tag{4.1}$$

Missile motion is achieved using a transition matrix and missile acceleration matrix and is specified by

$$M(k+1) = F_M M(k) + M_a(k)$$

$$= \begin{bmatrix} 1 & \Delta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_M(k) \\ \dot{x}_M(k) \\ y_M(k) \\ \dot{y}_M(k) \\ z_M(k) \\ \dot{z}_M(k) \end{bmatrix} + \begin{bmatrix} \dfrac{\ddot{x}_M(k)\Delta^2}{2} \\ \ddot{x}_M(k)\Delta \\ \dfrac{\ddot{y}_M(k)\Delta^2}{2} \\ \ddot{y}_M(k)\Delta \\ \dfrac{\ddot{z}_M(k)\Delta^2}{2} \\ \ddot{z}_M(k)\Delta \end{bmatrix}. \tag{4.2}$$

The missile transition matrix F_M translates the state vector from time k to $k+1$ assuming constant velocity. The missile acceleration matrix $M_a(k)$ accounts for deviation from a straight line trajectory due to gravity, parasitic and induced drag, engine thrust, and commanded guidance.

2. Commanded Acceleration Limiter

The AMRAAM missile is assumed to be capable of withstanding accelerations up to 52 times the acceleration of gravity g [10]. Broadston on Pehr placed a 30 g limit on each dimension of commanded guidance acceleration to prevent the simulated missile from exceeding this value [10], [11]. Since our commanded acceleration is applied as a vector in the x-y plane, we chose to limit PN commanded guidance acceleration magnitude to 50 g. This limit proved too high for DG guidance to be effective. Based on observed improvements in performance not seen with PN, the limit for DG was reduced to 15 g. This observation indicates DG is more sensitive to noise than PN guidance.

3. Thrust Characteristics

The missile thrust is 23,000 newtons for the first six seconds or "boost phase" of flight. This thrust is congruent with Broadston's six DOF model. Typical of most air-to-air missile models, thrust accelerates the missile at a constant rate until the boost phase is complete. The missile then coasts, expending kinetic energy and reducing speed due to aerodynamic drag.

C. TARGET MODEL

In this section we will explain how target motion is modeled. The target is modeled as a simple point mass in NED coordinates. The effects of drag and gravity are not modeled in the target. The target maintains a constant speed of Mach 0.83 and an altitude of 6000 meters throughout all three DOF simulations. It maintains a straight line trajectory until three seconds before impact. A six g turn is then initiated in the x-y plane and held until the simulation is complete.

1. Target Motion

Similar to the missile state vector, the target state vector $T(k)$, which describes the position and velocity of the missile at any given discrete time step k, is

$$T(k) = \begin{bmatrix} x_T(k) \\ \dot{x}_T(k) \\ y_T(k) \\ \dot{y}_T(k) \\ z_T(k) \\ \dot{z}_T(k) \end{bmatrix}. \tag{4.3}$$

Straight line target motion is generated using the target transition matrix F_T. This motion is unaffected by any acceleration and is specified by

$$T(k+1) = F_T T(k) = \begin{bmatrix} 1 & \Delta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_T(k) \\ \dot{x}_T(k) \\ y_T(k) \\ \dot{y}_T(k) \\ z_T(k) \\ \dot{z}_T(k) \end{bmatrix}. \tag{4.4}$$

2.   Target Turn

At three seconds to impact, the target begins a six g turn in the x-y plane. The target maintains a constant speed of Mach 0.83 and altitude of 6000 meters. Turn rate is also a constant due to unchanging acceleration and speed in the turn. Target angular turn rate ω_{turn} is [14]

$$\omega_{turn} = \frac{a_{turn}}{\sqrt{\dot{x}_T(k)^2 + \dot{y}_T(k)^2}}, \tag{4.5}$$

where target turn acceleration $a_{turn} = 6g$. The turning motion is generated by modifying the target transition matrix F_T to form [14]

$$T(k+1) = F_T T(k)$$

$$= \begin{bmatrix} 1 & \sin(\omega_{turn}\Delta)/\omega_{turn} & 0 & (1-\cos(\omega_{turn}\Delta))/\omega_{turn} & 0 & 0 \\ 0 & \cos(\omega_{turn}\Delta) & 0 & -\sin(\omega_{turn}\Delta) & 0 & 0 \\ 0 & (1-\cos(\omega_{turn}\Delta))/\omega_{turn} & 1 & \sin(\omega_{turn}\Delta)/\omega_{turn} & 0 & 0 \\ 0 & \sin(\omega_{turn}\Delta) & 0 & \cos(\omega_{turn}\Delta) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_T(k) \\ \dot{x}_T(k) \\ y_T(k) \\ \dot{y}_T(k) \\ z_T(k) \\ \dot{z}_T(k) \end{bmatrix} .(4.6)$$

D.   DRAG MODEL

Drag is modeled in this thesis as the composition of induced and parasitic drag. Induced drag models the effects of guidance acceleration and gravity while parasitic drag models the effects of missile shape and cross section in level flight. We utilize an imaginary "frame," which points in the direction of the missile's velocity vector. The combined forces due to induced and parasitic drag are converted to an acceleration based on the missile mass and given a vector direction opposite the velocity vector of the missile. We will discuss induced and parasitic forces separately before combining their effects and validating our drag model.

Drag is created by uneven pressures along the body of a missile [17]. The pressures on a missile change with the density of the air ρ and missile velocity v_M. Dynamic pressure Q, which is the function of missile velocity and altitude used to describe these changes, which impact both induced and parasitic drag, is [9]

$$Q = \frac{\rho \|v_M\|^2}{2}. \tag{4.7}$$

1.   Parasitic Drag

Parasitic drag accounts for friction based on the shape and cross section of a missile in level flight. Typical values for the parasitic drag coefficient C_{dp} are shown in Figure 5 [10]. It is clear the parasitic drag peaks at the sound barrier, Mach 1, and drops decidedly at supersonic speeds. Reduction in drag during boost phase due to the

aerodynamic effects of the plume at the tail of the missile is also evident. Force due to parasitic drag, which is proportional to the missile cross-sectional area S_{REF}, is [10]

$$F_{dp} = QC_{dp}S_{REF}.$$
(4.8)
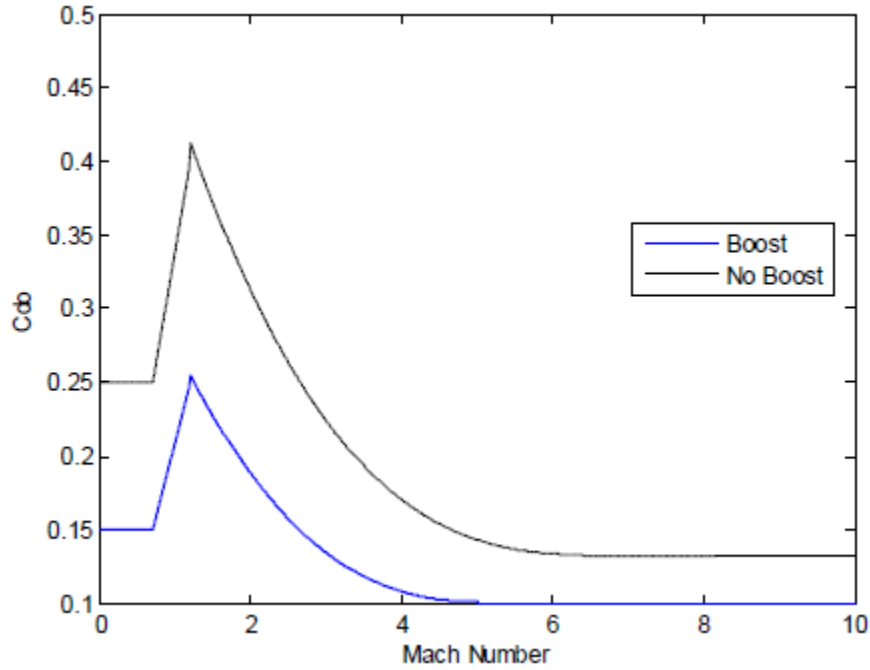
Figure 5.   Graph of the parasitic drag coefficient (from [10]).

2.   Induced Drag

Induced drag accounts for friction due to guidance acceleration and gravity. Typically induced drag is calculated using angles of attack. Due to the point mass nature of this simulation, we assume the missile is pointing in the direction of its velocity vector. Induced drag is determined using normal forces in azimuth and elevation acting on the body [10]. Gravity is accounted for directly in the elevation component of induced drag. The normal induced drag forces in the azimuth F_{ia} and elevation F_{ie} directions are [10]

$$F_{ia} = m(a_{Ma})$$
(4.9)

and

$$F_{ie} = m(a_{Me} - g),\qquad(4.10)$$

where a_{Ma} and a_{Me} are the missile azimuth and elevation commanded accelerations respectively, and m is the missile mass. Missile commanded accelerations are determined by the guidance law.

The normal forces are converted to azimuth and elevation induced drag coefficients, [10]

$$C_{na} = \frac{F_{ia}}{QS_{REF}}\qquad(4.11)$$

and

$$C_{ne} = \frac{F_{ie}}{QS_{REF}}.\qquad(4.12)$$

These coefficients are combined to determine the overall induced drag coefficient [10]

$$C_{di} = \frac{\left(C_{na}^2 + C_{ne}^2\right)}{\pi(e)AR}.\qquad(4.13)$$

The wing efficiency relative to an elliptical planform e and wing aspect ratio AR are constants drawn from Broadston and Pehr [11].

At subsonic speed the overall induced drag coefficient is based on the highest subsonic parasitic drag coefficient, $C_{dp} = 0.25$, and the applied force. Broadston makes this assumption noting it is a rough approximation that has a small impact for a short period of time at the beginning and end of the simulation [11]. The subsonic overall induced drag coefficient is [2]

$$C_{di} = 0.25\frac{\sqrt{F_{ia}^2 + F_{ie}^2}}{mg}.\qquad(4.14)$$

The induced drag coefficient encompasses the effect of gravity and all normal forces on the missile. The total induced drag F_{di}, which incorporates dynamic pressure and the shape of the missile, is [10]

33

$$F_{di} = QC_{di}S_{REF}. \tag{4.15}$$

3. Total Drag

Total force due to drag is proportional to dynamic pressure, the missile cross-sectional area, as well as the coefficients of parasitic and induced drag. It is converted to an acceleration based on the missile mass and given a vector direction opposite the velocity vector of the missile. The magnitude of acceleration due to drag is

$$\left\| a_{Mdrag} \right\| = \frac{\left(F_{dp} + F_{di} \right)}{m}. \tag{4.16}$$

4. Drag Model Validation

We compared the three DOF and six DOF model maximum missile range to validate our drag model. To ensure an accurate comparison, a stationary target was used and the missile was allowed to travel until its speed reached zero. The difference in effective range is only a result of the drag forces experienced by the missile in flight. Open source data for the AIM-120D indicates a maximum range of 72 kilometers [10]. Pehr determined the maximum range of his drag model to be 76.6 kilometers [10]. The three DOF model yielded a maximum range of 77.2 kilometers. This is within roughly seven percent of the open source data and one percent of Pehr's result. Based on this result we consider the three DOF model to be validated as a useful simplified version for Kalman filter implementation.

E. NOISE MODEL

The noise in this thesis is more accurately represented than in previous work by Broadston and Pehr. They used a "direct insertion" method, adding noise with a constant standard deviation into the actual range, range rate, LOS angle and LOS angle rate values [11]. The simulated missile in our model is equipped with range and LOS angle sensors only. LOS angle rate and range rate are extrapolated from these inputs.

Noisy LOS angle and range measurements in the three DOF model are generated with

$$\theta_{L_{nz}} = \theta_L + \sigma_{\theta_L} n_{rand} \tag{4.17}$$

and

$$r_{nz} = r + \sigma_r n_{rand}, \tag{4.18}$$

where n_{rand} is a pseudorandom value, drawn from a zero mean Gaussian distribution with variance of one. The simulated sensor standard deviations, which adjust the Gaussian variance, are

$$\sigma_{\theta_L} = f_{noise} \sigma_{\theta_L\_base} \tag{4.19}$$

and

$$\sigma_r = f_{noise} \sigma_{r\_base}, \tag{4.20}$$

where f_{noise} is a multiplier used to systematically increase noise, and subsequently Kalman filter measurement covariance $R(k)$, when desired. For the three DOF model, the baseline range and LOS angle simulated sensor standard deviations, σ_{r_base} and $\sigma_{\theta_L\_base}$ are defined as 10 meters and 1 mrad as established by Pehr [10]. Baseline noise, where $f_{noise} = 1$, is used in all simulations with the exception of the maximum sustainable noise study. Due to the Gaussian nature of the noise, approximately 70 percent of the noisy measurements will be within one standard deviation of the actual LOS angle or range value.

A small standard deviation can have a significant effect on the maximum effective range of a missile. Measurement noise is interpreted by the missile guidance system as target acceleration. The guidance system responds to noise with slight guidance accelerations. These missile accelerations produce an induced drag, which increases exponentially as missile speed approaches Mach 1. Small unnecessary guidance accelerations over the course of the entire missile flight can produce dramatic performance deterioration on the order of several kilometers.

F.   FILTER IMPLEMENTATION

In this section we will discuss the four filters of our three DOF model. They are divided into PN and DG filters, each guidance law having a range and bearing filter. The division stems from the nature of the laws themselves. Simple two dimension filters suffice for PN because it requires only LOS angle rate and range rate. The target state parameters used in DG guidance requires a more complicated three dimension filter. While it would be simpler to make both PN and DG operate with a three dimension filter, a higher dimension filter introduces more extrapolation and should only be used when necessary.

Each guidance law uses two Kalman filters to obtain the parameters necessary for guidance implementation. The first filter estimates the LOS angle state, the second estimates range state. These filters estimate time derivatives of LOS angle and range based on noisy measurements received from the simulated missile sensors. Unique aspects of the two types of filters include: the state estimate, state estimate covariance, process covariance, and deterministic inputs. We will examine the PN and DG differences in each of these components one at a time.

1.   State Estimate

The state estimate summarizes a deterministic system [15]. At any discrete time $k+1$ the state estimate can be determined by (3.7). A Kalman filter estimates the state using the process described in Chapter III. Initial state estimates in this thesis are populated with the actual parameter values. This method assumes the launching platform has a tracking system accurate enough to make its noise negligible compared to the missile's sensor noise.

The PN guidance law only requires the missile lead angle η_M, LOS angle rate $\dot{\theta}_L$ and closing speed v_c. Since the missile state is assumed to be known, an estimate of η_M is also known based on the estimated LOS angle. The LOS angle rate $\dot{\theta}_L$ and closing speed v_c are estimated using two dimensional LOS angle and range states

36

$$\hat{x}_{\theta_L}(k) = \begin{bmatrix} \hat{\theta}(k) \\ \hat{\dot{\theta}}(k) \end{bmatrix} \qquad (4.21)$$

and

$$\hat{x}_r(k) = \begin{bmatrix} \hat{r}(k) \\ \hat{\dot{r}}(k) \end{bmatrix}. \qquad (4.22)$$

The DG guidance law is more complex. To implement DG the following information is required:

- target lead angle η_T, defined as the angle between the target's velocity vector and the LOS.

- missile lead angle η_M, defined as the angle between the missile's velocity vector and the LOS.

- magnitude of target acceleration $\|a_T\|$.

- LOS angle rate of change $\dot{\theta}_L$.

- closing speed v_c.

To provide these parameters, DG filters use three dimensional state vectors

$$\hat{x}_{\theta_L}(k) = \begin{bmatrix} \hat{\theta}_L(k) \\ \hat{\dot{\theta}}_L(k) \\ \hat{\ddot{\theta}}_L(k) \end{bmatrix} \qquad (4.23)$$

and

$$\hat{x}_r(k) = \begin{bmatrix} \hat{r}(k) \\ \hat{\dot{r}}(k) \\ \hat{\ddot{r}}(k) \end{bmatrix}. \qquad (4.24)$$

Estimates of target acceleration magnitude $\|\hat{a}_T\|$ and lead angle $\hat{\eta}_T$ are calculated using the third dimension time derivative in a process described later in this chapter.

2. State Estimate Covariance

The state estimate covariance matrix $P(k)$ reflects the accuracy of the filter's state estimate at time k. The diagonal elements display the variance of each element in the state estimate. This implies the size of the state estimate covariance is determined by the dimensions of the state estimate. The state estimate covariance matrix is initialized with the covariance of the simulated missile's onboard sensors. Over several time steps, the filter mitigates the effect of noise and creates more accurate estimates with a lower covariance. State covariance generally continues to decline to a steady state value as more measurements are taken, even if the target maneuvers. Initial two dimensional PN filter covariance matrices are [16]

$$P_{\theta_L}(0) = \begin{bmatrix} \sigma_{\theta_L}^2 & 0 \\ 0 & \dfrac{2\sigma_{\theta_L}^2}{\Delta^2} \end{bmatrix} \tag{4.25}$$

and

$$P_r(0) = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \dfrac{2\sigma_r^2}{\Delta^2} \end{bmatrix}. \tag{4.26}$$

Initial three dimensional DG filter covariance matrices are [16]

$$P_{\theta_L}(0) = \begin{bmatrix} \sigma_{\theta_L}^2 & 0 & 0 \\ 0 & \dfrac{2\sigma_{\theta_L}^2}{\Delta^2} & 0 \\ 0 & 0 & \dfrac{4\sigma_{\theta_L}^2}{\Delta^4} \end{bmatrix} \tag{4.27}$$

and

$$P_r(0) = \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \dfrac{2\sigma_r^2}{\Delta^2} & 0 \\ 0 & 0 & \dfrac{4\sigma_r^2}{\Delta^4} \end{bmatrix}. \tag{4.28}$$

3. Process Covariance

Process covariance $Q(k)$ is the covariance of process noise, the zero mean white Gaussian process noise used to describe unpredictable state variations. Process covariance for two dimensional PN filters is [16]

$$Q(k) = q^2(k) \begin{bmatrix} \dfrac{\Delta^3}{3} & \dfrac{\Delta^2}{2} \\ \dfrac{\Delta^3}{2} & \Delta \end{bmatrix}. \tag{4.29}$$

Process covariance for three dimensional DG filters is [16]

$$Q(k) = q^2(k) \begin{bmatrix} \dfrac{\Delta^5}{20} & \dfrac{\Delta^4}{8} & \dfrac{\Delta^3}{6} \\ \dfrac{\Delta^4}{8} & \dfrac{\Delta^3}{3} & \dfrac{\Delta^2}{2} \\ \dfrac{\Delta^3}{6} & \dfrac{\Delta^2}{2} & \Delta \end{bmatrix}. \tag{4.30}$$

The process covariance weighting factor $q^2(k)$ can be seen in (4.29) and (4.30). This weighting factor can be "tuned" prior to and during the simulation to improve missile performance. The weighting factor is determined by setting the plant covariance equal to the covariance of a state variable due to the most severe target maneuver. This can be done with any of the state variables, but higher dimensions are more conservative [16]. By using the most severe target maneuver, we are tuning the weighting factor to follow the target through any possible maneuver. These weighting factors are described individually.

a. LOS Angle Weighting Factor

The LOS angle weighting factor $q_{\theta_L}{}^2(k)$ for both DG and PN filters is determined using the covariance of the LOS angle rate. The change in LOS angle θ_L is [14]

$$\hat{\theta}_L(k+1) - \hat{\theta}_L(k) = \tan^{-1}\left(\frac{\left\|\hat{a}_{Tperp}(k)\right\|\Delta^2}{2\hat{r}(k)}\right) \approx \frac{\left\|\hat{a}_{Tperp}(k)\right\|\Delta^2}{2\hat{r}(k)}, \qquad (4.31)$$

where $\left\|\hat{a}_{Tperp}\right\|$ is the estimated magnitude of target acceleration perpendicular to the LOS. An approximation can be made since the change in LOS angle is very small in one time step. We consider the change in LOS angle to be the standard deviation of the scenario with the hardest feasible target turn perpendicular to the LOS if $\left\|\hat{a}_{Tperp}\right\| = 6g$. By taking the square of this standard deviation we attain the covariance. Setting the two covariance values equal to each other yields [14]

$$\left(\hat{\theta}(k+1) - \hat{\theta}(k)\right)^2 = q_{\theta_L}^2(k)\frac{\Delta^3}{3}. \qquad (4.32)$$

Substituting the time derivative of (4.31) into (4.32) and solving for the LOS angle weighting factor yields [14]

$$q_{\theta_L}^2(k) = \frac{3\left\|\hat{a}_{Tperp}(k)\right\|^2 \Delta}{4\hat{r}(k)^2}. \qquad (4.33)$$

It is important to note, the LOS angle weighting factor is dependent on range. As the missile closes with the target, target accelerations will affect a larger change in LOS angle. Since the weighting factor is range dependent, it will increase in size to compensate for the larger change in LOS angle.

The LOS angle weighting factor is also manipulated to improve DG guidance performance and combat noise sensitivity. During flight toward the target, the LOS angle weighting factor is multiplied by 10 for DG guidance. This helped the filter track the target while limiting noise in commanded guidance acceleration and a corresponding drop in speed. When the missile reaches a range of 10 kilometers, the LOS angle weighting factor is multiplied by 100. The higher process covariance in the end game helps the filter account for terminal target maneuvers. Increasing the weighting factor at 10 kilometers is logical because a target maneuver before this point is unlikely. At distant

ranges the missile can respond to target maneuvers with small course adjustments. A target maneuver is not likely until the missile is relatively close, which maximizes the strain on the missile guidance.

b. *Range Weighting Factor*

The range weighting factor $q_r^{\ 2}(k)$ for both DG and PN filters is determined using the covariance of the range rate. The range state estimation is not sensitive to range. This means the range weighting factor is constant throughout the scenario. A change in range r is [14]

$$\hat{r}(k+1)-\hat{r}(k)=\frac{\left\|\hat{a}_{Tpara}(k)\right\|\Delta^2}{2}, \tag{4.34}$$

where $\left\|\hat{a}_{Tpara}\right\|$ is the estimated magnitude of target acceleration parallel to the LOS. We consider the change in range to be the standard deviation of the scenario with the hardest feasible target turn parallel to the LOS if $\left\|\hat{a}_{Tpara}\right\|=6g$. Squaring the standard deviation and setting the two covariance values equal to each other yields [14]

$$\left(\hat{r}(k+1)-\hat{r}(k)\right)^2=q_r^{\ 2}(k)\frac{\Delta^3}{3}. \tag{4.35}$$

Substituting the time derivative of (4.34) into (4.35), and solving for the range weighting factor yields [14]

$$q_r^{\ 2}(k)=\frac{3\left\|\hat{a}_{Tpara}(k)\right\|^2\Delta}{4}. \tag{4.36}$$

4. Deterministic Inputs

The deterministic input vector $u(k)$ accounts for known changes in the state due to missile motion, namely accelerations due to missile drag and thrust. Known changes in the position, velocity, and acceleration of the state are added to the predicted state

estimate. The performance of the filter is improved when a deterministic input is provided to the predicted state estimation in (3.7) because the amount of motion being estimated is reduced.

The directions of missile thrust and drag are opposite each other but parallel to the missile velocity vector. The deterministic input is [14]

$$a_{Mu} = a_{Mthrust} - a_{Mdrag} .$$
(4.37)

The portion of the deterministic input perpendicular to the LOS affects the LOS angle state estimation and the parallel portion affects the range state estimation.

The PN and DG filters have similar deterministic inputs with one exception. We model target acceleration as process noise, so adding the deterministic acceleration of the missile would drastically increase the noise in the filter. We must wait until after the filter generates a corrected state estimate to add the deterministic acceleration of the missile. Let us first examine the LOS angle and then range deterministic inputs.

a.   *LOS Angle Deterministic Input*

The LOS angle state estimation is affected by accelerations perpendicular to the LOS. The exact change in LOS angle θ_L is determined as described in (4.31). The exact change in the LOS angle rate $\dot{\theta}_L$ is [14]

$$\hat{\dot{\theta}}_L(k+1) - \hat{\dot{\theta}}_L(k) = \frac{4a_{Muperp}(k)\hat{r}(k)\Delta}{\left(4\hat{r}(k)^2 + a_{Muperp}(k)^2\Delta^4\right)} .$$
(4.38)

Combining (4.31) and (4.38) gives the estimated PN LOS angle deterministic input [14]

$$\hat{u}_{\theta_L}(k) = \begin{bmatrix} \tan^{-1}\left(\dfrac{a_{Muperp}(k)\Delta^2}{2\hat{r}(k)}\right) \\ \dfrac{4a_{Muperp}(k)\hat{r}(k)\Delta}{\left(4\hat{r}(k)^2 + a_{Muperp}(k)^2\Delta^4\right)} \end{bmatrix} .$$
(4.39)

Expanding (4.39) to three dimensions gives the estimated DG LOS angle deterministic input [14]

$$\hat{u}_{\theta_L}(k) = \begin{bmatrix} \tan^{-1}\left(\dfrac{a_{Muperp}(k)\Delta^2}{2\hat{r}(k)}\right) \\[2ex] \dfrac{4a_{Muperp}(k)\hat{r}(k)\Delta}{\left(4\hat{r}(k)^2 + a_{Muperp}(k)^2\Delta^4\right)} \\[2ex] a_{Muperp}(k) \end{bmatrix}. \tag{4.40}$$

The acceleration portion of the DG LOS angle deterministic input will not be added to the predicted state estimate of (3.7). The altered predicted state estimate is

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k) \begin{bmatrix} \tan^{-1}\left(\dfrac{a_{Muperp}(k)\Delta^2}{2\hat{r}(k)}\right) \\[2ex] \dfrac{4a_{Muperp}(k)\hat{r}(k)\Delta}{\left(4\hat{r}(k)^2 + a_{Muperp}(k)^2\Delta^4\right)} \\[2ex] 0 \end{bmatrix}. \tag{4.41}$$

Deterministic acceleration is added to the corrected state estimate of (3.13), which yields

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)\nu(k+1) + \begin{bmatrix} 0 \\ 0 \\ a_{Muperp}(k) \end{bmatrix}. \tag{4.42}$$

b.   *Range Deterministic Input*

The range state estimation is affected by accelerations parallel to the LOS. Acceleration parallel to the LOS is directly proportional to the change in range r and range rate $\dot{r}$. The estimated PN range deterministic input is [14]

$$u_r(k) = a_{Mupara}(k) \begin{bmatrix} \dfrac{\Delta^2}{2} \\ \Delta \end{bmatrix}. \tag{4.43}$$

Expanding (4.43) to three dimensions gives the estimated DG range deterministic input [14]

$$u_r(k) = a_{Mupara}(k) \begin{bmatrix} \dfrac{\Delta^2}{2} \\ \Delta \\ 1 \end{bmatrix}. \tag{4.44}$$

Corresponding altered predicted and corrected state estimate equations are

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)a_{Mupara}(k) \begin{bmatrix} \dfrac{\Delta^2}{2} \\ \Delta \\ 0 \end{bmatrix} \tag{4.45}$$

and

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W(k+1)v(k+1) + a_{Mupara}(k) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.46}$$

G.   GUIDANCE LAW IMPLEMENTATION

The DG guidance law proved to be much more difficult to implement than PN guidance. Simple two dimensional filters provide direct estimates for the required parameters to implement the PN guidance law. Implementation of DG guidance requires knowledge of the magnitude of the target's acceleration $\|a_T\|$ and lead angle η_T. Estimates of these two parameters, and subsequently DG guidance, are more sensitive to noise due to the extrapolation involved.

The two target parameters required for DG guidance are estimated from available LOS information. Using three dimensional filters for DG we obtain estimates of the LOS angle acceleration $\hat{\ddot{\theta}}_L$ and range acceleration $\hat{\ddot{r}}$. Since we assume the missile state vector is known, we can remove the effects of the missile, leaving only the effects of the target. Using the remainders of LOS angle acceleration and range acceleration, we estimate the target's magnitude of acceleration $\|\hat{a}_T\|$ and lead angle $\hat{\eta}_T$.

The process we use to estimate the target parameters is different for DG guidance under noiseless and Kalman filtered noise conditions. To remove the effects of the missile

on the LOS, missile guidance acceleration must be known. The missile guidance acceleration is calculated using Kalman filter outputs. It is not available to the Kalman filter for estimation of target parameters in the same discrete time step. For estimates of target parameters in noiseless DG scenarios, we use the previous time step missile guidance acceleration and the current time step value of the remaining missile state parameters. Since noiseless scenarios have smooth guidance acceleration, the change is small from one time step to the next, which makes this method effective. For scenarios with Kalman filtered noise, guidance acceleration is not smooth. We use the previous time step for all missile state parameters that makes the calculation for guidance acceleration accurate and less erratic, but implements the guidance a time step late. To further smooth the commanded guidance acceleration, we average the current time step value with the previous three discrete time steps. These techniques, in conjunction with a maximum guidance acceleration limit, balance stability and responsiveness to effectively implement DG guidance.

In Chapter IV, we examined the simulation methodology of the three DOF model. We built an understanding of the kinematic boundary and its relation to our simulation. Missile and target motion including the effects of drag were generated. Our four Kalman filters were described by their state estimate, state covariance estimate, process covariance, and deterministic inputs. Finally, implementation of PN and DG guidance laws were discussed. Using this three DOF model numerous simulations were run to investigate the objectives of this thesis:

- Determine the effect of Kalman filtered noise on the performance of PN and DG guidance laws.

- Determine if DG provides a tactical advantage over PN for guidance of the simulated AMRAAM in the presence of Kalman filtered noise.

- Determine what level of Kalman filtered noise these guidance laws can withstand.

- Explore the effect of discrete time measurement sample rate on guidance law performance.

Chapter V summarizes the results and analysis of our simulations in pursuit of these objectives.

THIS PAGE INTENTIONALLY LEFT BLANK

V.   SIMULATION RESULTS AND ANALYSIS

Several tests are completed in order to meet our objectives. Tests A (PN) and B (DG) compare individual guidance law performance under noiseless and Kalman filtered noise conditions. Test C compares PN against DG performance with Kalman filtered noise to determine any improvement in guidance of the simulated AMRAAM. Test D compares individual guidance law noise sensitivity at a test range 10 percent below the corresponding law's noiseless maximum range. Noise sensitivity is measured by determining the maximum noise factor f_{noise} each law can withstand while still maintaining 70 percent efficiency. Test E compares individual guidance law performance at a baseline discrete time step size of 10 milliseconds with its performance at half and twice the baseline rate to determine sample rate sensitivity.

The kinematic boundary in simulations with Kalman filtered noise represents the maximum range from which the missile reaches the kill radius in at least 14 of 20 simulations. Defining the maximum range in this manner maintains the 70 percent efficiency standard set by Pehr [10]. As with any Gaussian white noise process, statistical outliers may exist. Kinematic boundaries with a one degree resolution provide 180 opportunities for an outlier to appear. The number of simulations was chosen to minimize statistical outliers and keep the time required to generate a kinematic boundary below 72 hours. Kinematic boundaries involving greater than 20 simulations are more statistically robust but exponentially time intensive.

Instability in the aft quadrant appears in all kinematic boundaries in this thesis. Kinematic boundaries can be reproduced for noiseless simulations but not for Kalman filtered noise simulations. Due to the pseudorandom nature of the Kalman filtered noise, the instability in the aft quadrant is slightly different with each simulation. This can generate aft quadrant results in which maximum effective range with Kalman filtered noise is larger than noiseless range. The effects of particular simulation parameters on this instability are discussed as they present themselves in test results. Our hypothesis for the cause of this instability is presented in Chapter VI.

47

A. TEST A—PN: NOISELESS VERSUS FILTERED NOISE

The PN guidance law has a low sensitivity to Kalman filtered noise. The kinematic boundaries for PN performance under noiseless and Kalman filtered noise conditions are shown in Figure 6. Due to the Gaussian nature of the noise, approximately 70 percent of the noisy measurements will be within 10 meters or 1 mrad of the actual LOS angle or range value. The kinematic boundaries are practically identical with the exception of some instability in the aft quadrant. A plot of the maximum effective range for a 180 degree arc is shown in Figure 7. The range difference between the kinematic boundaries is shown in Figure 8. Excluding instability in the aft quadrant, slight constant noise degradation throughout the 360 degree view of about one kilometer can be seen.
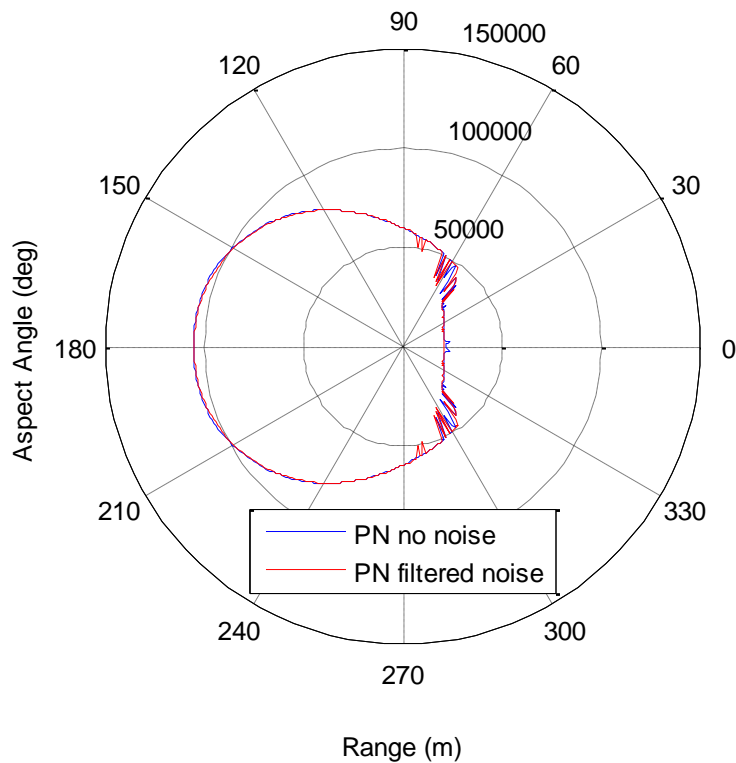
Figure 6.   Kinematic boundaries for PN guidance under noiseless and Kalman filtered noise conditions.
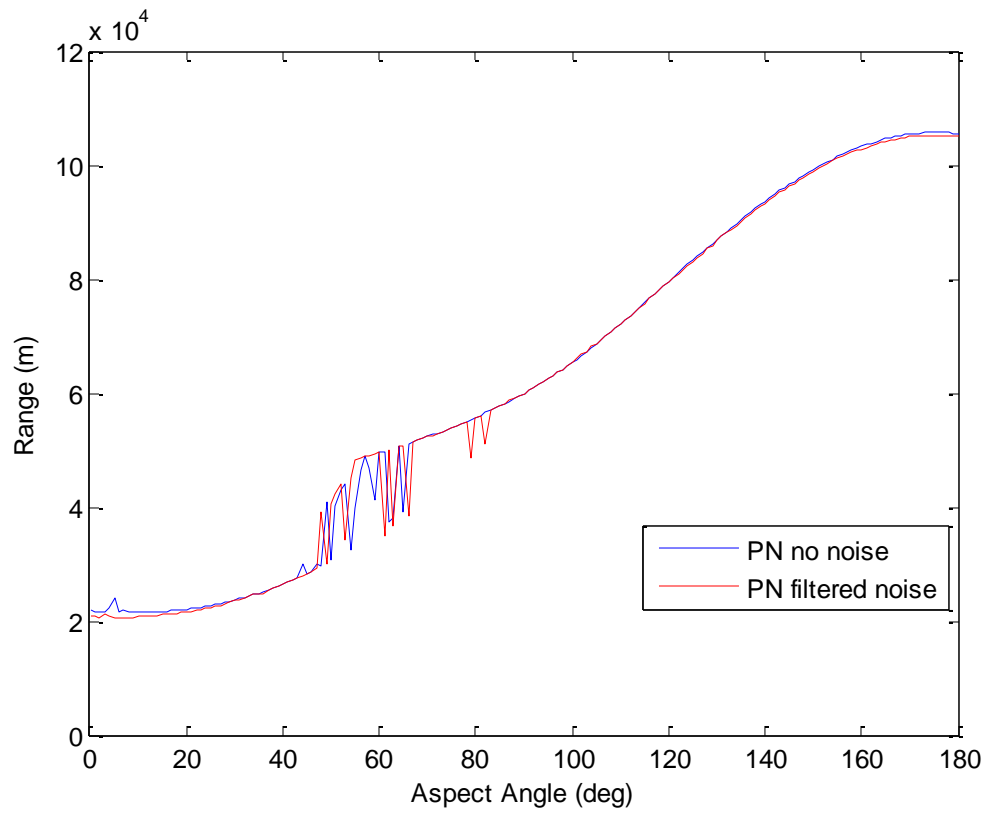
Figure 7.   Maximum effective range of PN guidance under noiseless and Kalman
filtered noise conditions.
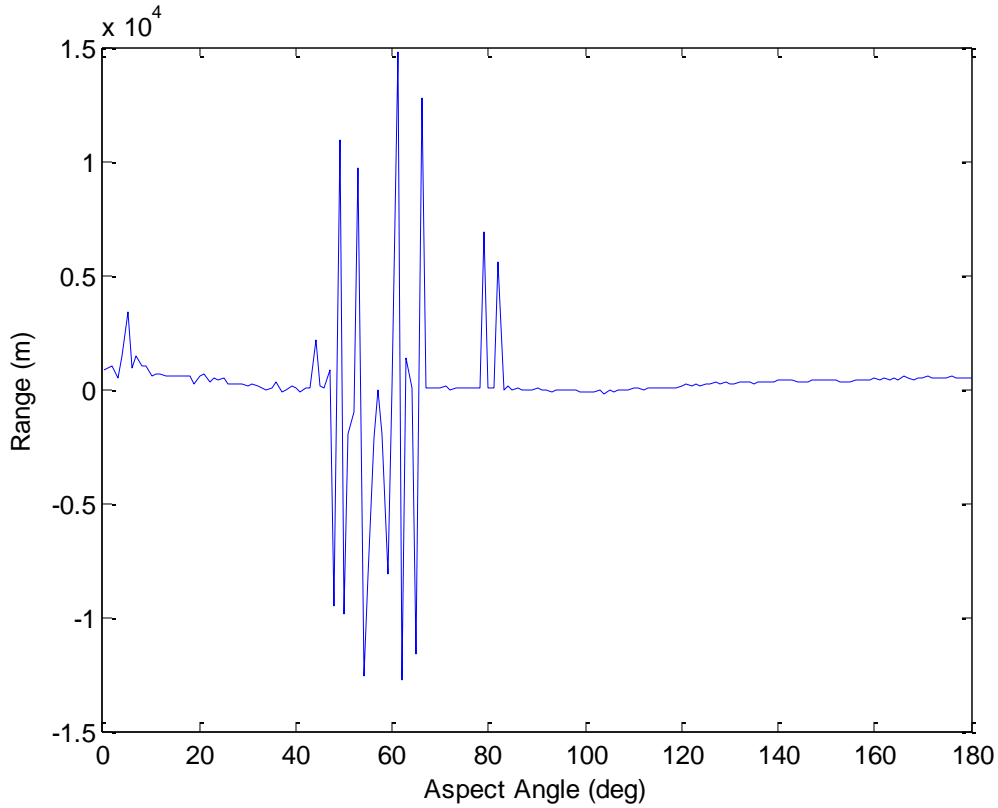
Figure 8.   Difference plot of PN maximum effective range under noiseless and
Kalman filtered noise conditions.

B.   TEST B—DG: NOISELESS VERSUS FILTERED NOISE

The DG guidance law has a more pronounced sensitivity to Kalman filtered noise than PN guidance. The kinematic boundaries for DG performance under noiseless and Kalman filtered noise conditions are shown in Figure 9. The Kalman filtered noise kinematic boundary is noticeable shorter in almost all aspect angles and experiences a more pronounced instability in the aft quadrant. This instability may be compounded by the extreme noise sensitivity of DG target state estimates. A plot of the maximum effective range for a 180 degree arc is shown in Figure 10. The range difference between the kinematic boundaries is shown in Figure 11. Excluding the instability in the aft quadrant, noise degradation throughout the 360 degree view varies from one to six kilometers.
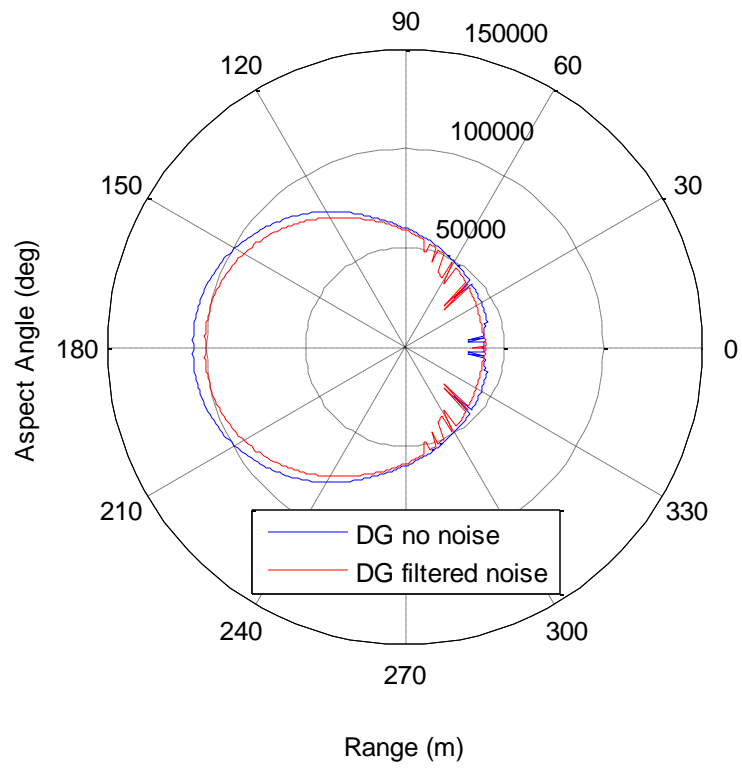
Figure 9. Kinematic boundaries for DG guidance under noiseless and Kalman filtered noise conditions.
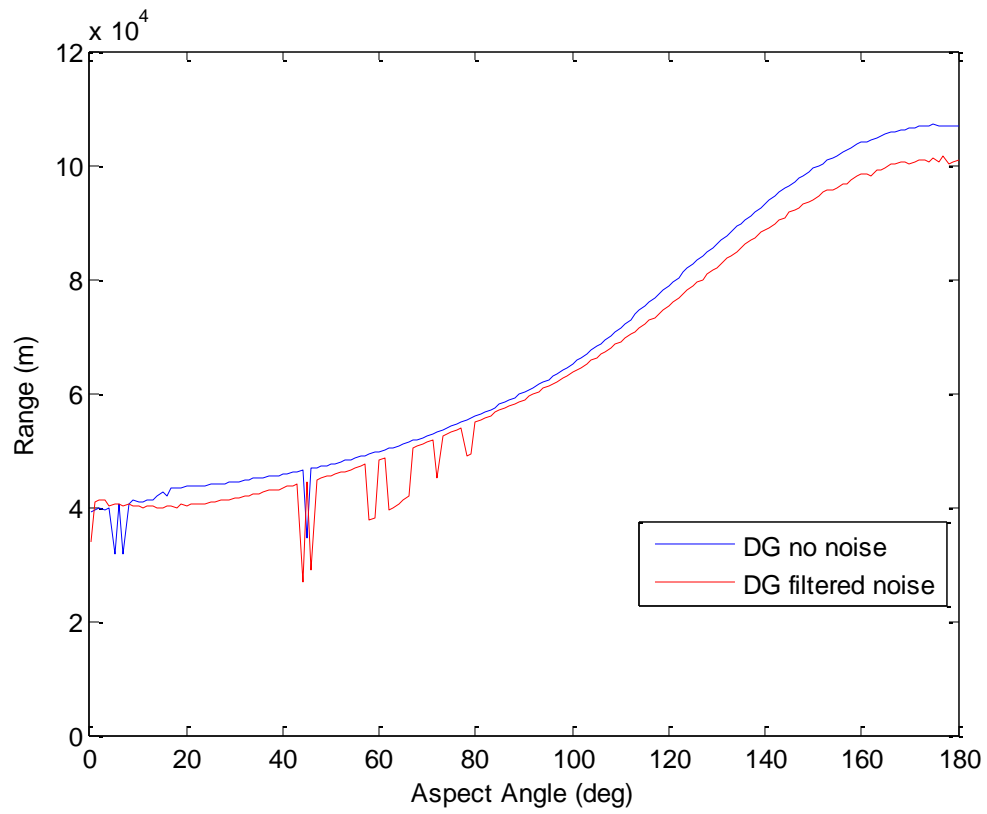
Figure 10.   Maximum effective range of DG guidance under noiseless and Kalman filtered noise conditions.
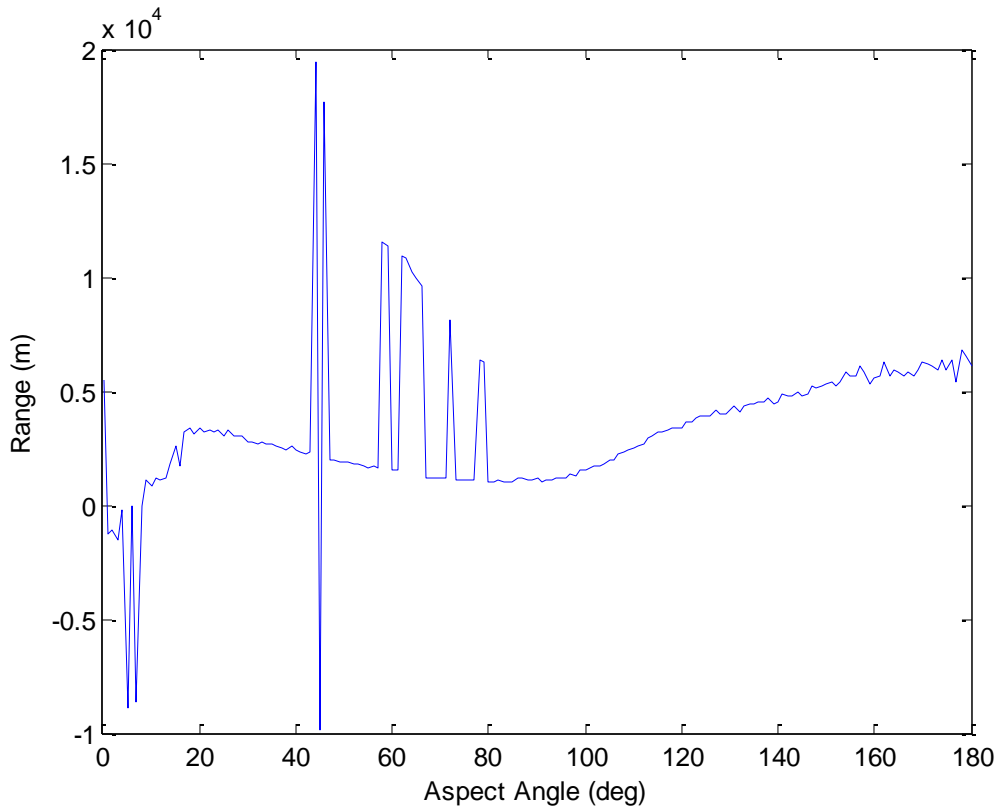
Figure 11.   Difference plot of DG maximum effective range under noiseless and Kalman filtered noise conditions.

C.   TEST C—KALMAN FILTERED NOISE: DG VERSUS PN

The DG guidance law provides a tactical advantage over PN for guidance of the simulated AMRAAM. While DG guidance suffers slightly larger noise degradation in most aspect angles, it performs much better than PN in tail chase scenarios. The effective range advantage in the aft quadrant outweighs the disadvantage in the forward quadrant. The kinematic boundaries for DG and PN performance with Kalman filtered noise are shown in Figure 12. The difference between the kinematic boundaries in the aft quadrant is clear. A plot of the maximum effective range for a 180 degree arc is shown in Figure 13. The range difference plot, shown in Figure 14, highlights the improvement in guidance of the simulated AMRAAM. The DG guidance law provides a maximum effective range that extends between 17 to 20 kilometers beyond PN in the aft quadrant while sacrificing roughly five kilometers in the remaining aspect angles. The effective

range advantage in the aft quadrant outweighs the disadvantage in the other quadrants. These results serve as a point of departure for research of a hybrid combination of PN and DG guidance laws.
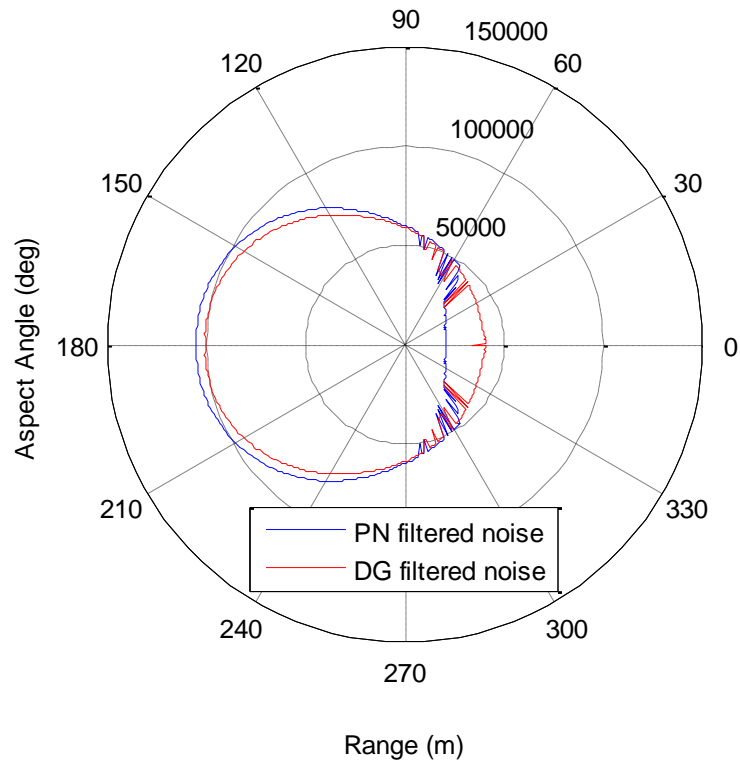
Figure 12.   Kinematic boundaries for PN and DG guidance with Kalman filtered noise.
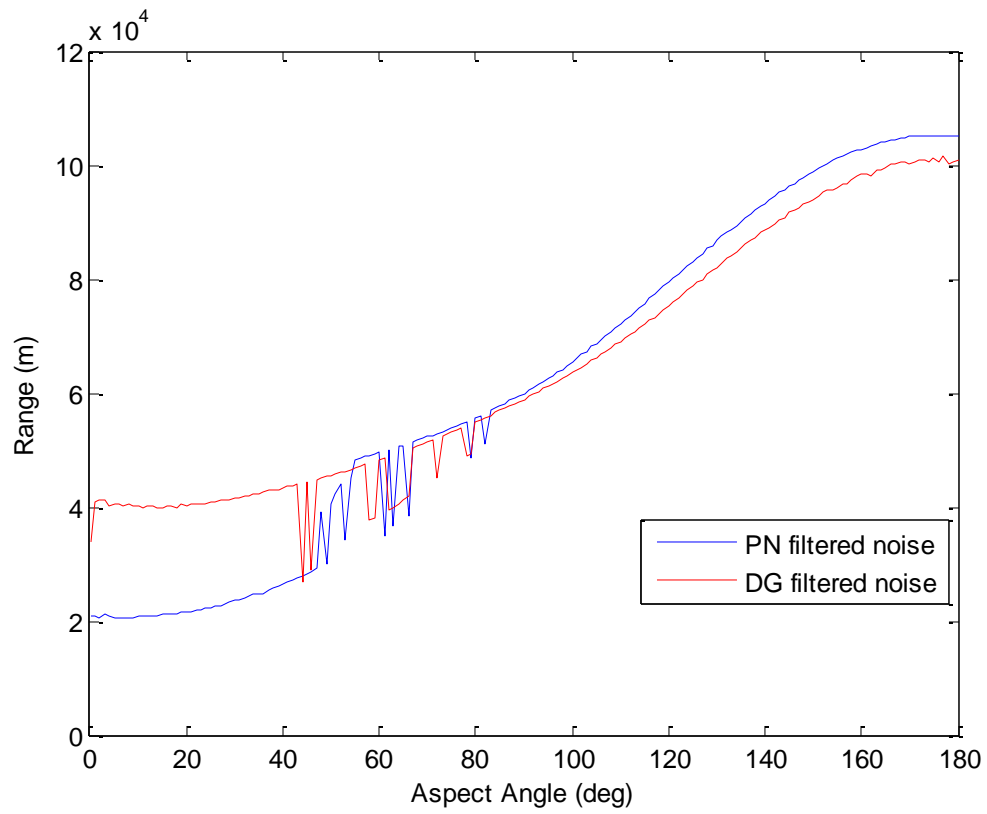
Figure 13.   Maximum effective range of PN and DG guidance with Kalman filtered noise.
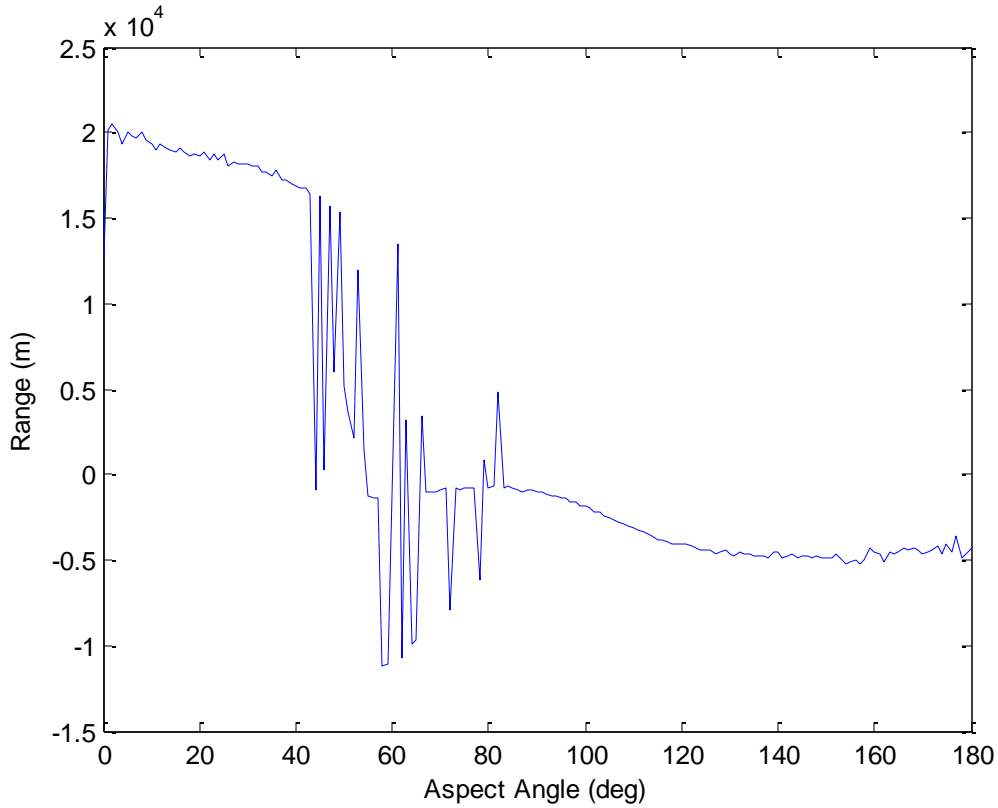
Figure 14.   Difference plot of PN and DG maximum effective range with Kalman filtered noise.

D.   TEST D—MAXIMUM SUSTAINABLE NOISE

Noise sensitivity is measured by determining the maximum noise factor f_{noise} each law can withstand while still scoring a hit in at least 14 out of 20 simulations. The noise factor is a direct multiplier of the simulated sensor baseline standard deviations as seen in (4.19) and (4.20). All simulations for Test D are conducted with Kalman filtered noise at a range 10 percent below the corresponding law's maximum effective range with no noise.

Plots of the maximum sustainable noise factor for PN and DG guidance are shown in Figure 15 and Figure 16 respectively. The PN noise factor appears to be quite sensitive to the aspect angle while the DG noise factor presents as a relatively smooth curve. Both guidance laws appear to be more sensitive in the aft quadrant than the forward quadrant. The PN noise factor shows less sensitivity than DG in the aft quadrant

56

but appears to oscillate every 30 degrees in the forward quadrant. The spike in DG noise factor at exactly 45 degrees appears to be a statistical outlier. Early versions of this test failed to produce the same spike at 45 degrees. Closer analysis also reveals the spike is a single data point and values at 44 and 46 degrees support the gradual curve of the data.

A noise factor below one indicates the guidance was only effective at the test range with less than the baseline Kalman filtered noise. Several data points of zero noise factor for PN guidance appear in Figure 15. The maximum noise factor test range is 10 percent below the noiseless maximum effective range. The accuracy of the noise factor being tested is 0.1. A maximum noise factor of zero indicates PN guidance was unable to maintain 70 percent effectiveness at the test range with a noise factor of 0.1. The aspect angles that produce these zero values correspond to regions of instability. Zero values inside the region of instability are to be expected; differences between noiseless and noisy simulations can be on the order of 15 kilometers, as shown in Figure 8.
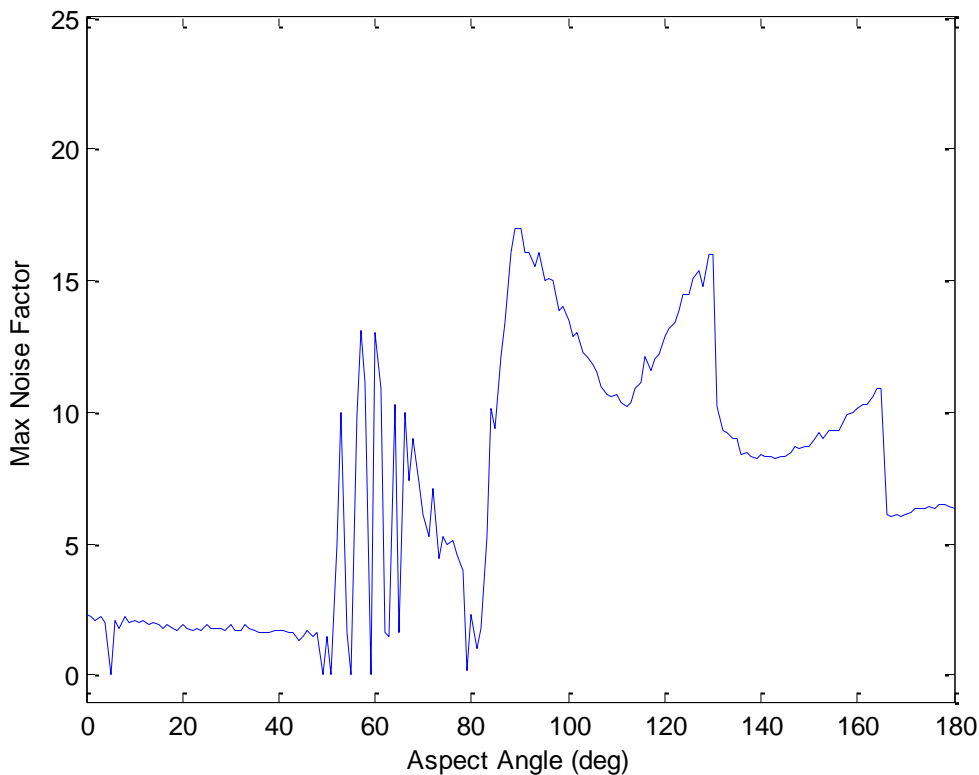
Figure 15.   Maximum noise factor for PN guidance.
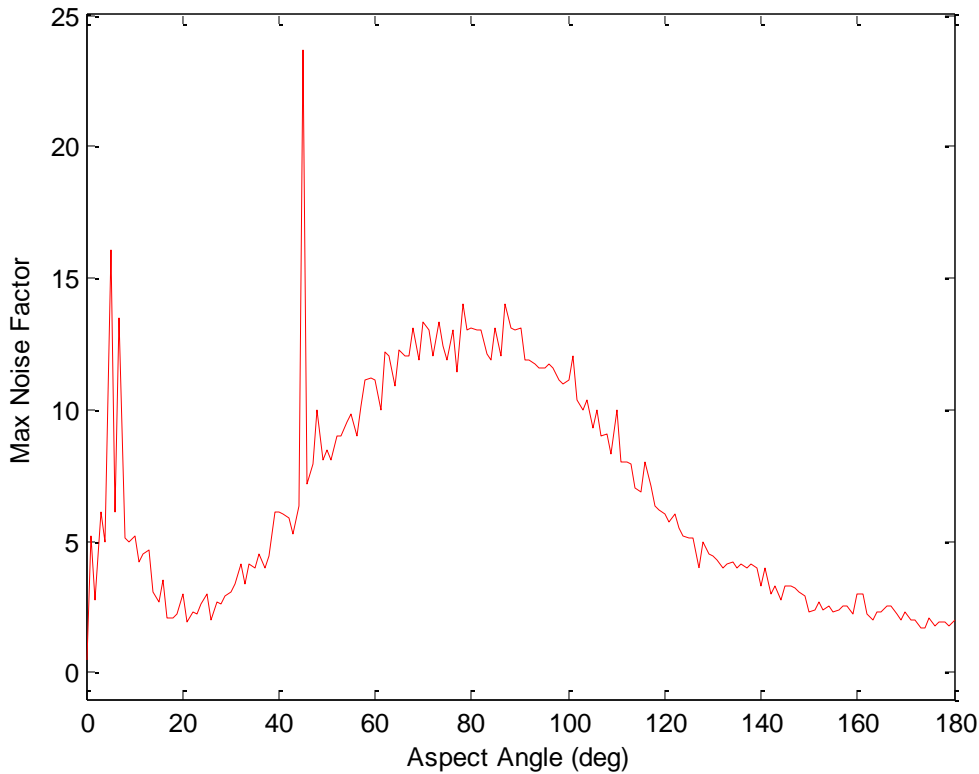
Figure 16.   Maximum noise factor for DG guidance.

E.   TEST E—SAMPLE RATE SENSITIVITY

Changing the discrete time sample rate affects the response time of the missile guidance system regardless of the guidance law employed. We expect performance to improve with smaller rates and deteriorate with larger rates. The baseline rate of 10 milliseconds was chosen for the simulated AMRAAM. Sample rates of half and twice the baseline were chosen for the sensitivity study as results for larger sample intervals were excessively poor.

Kinematic boundaries for PN performance with Kalman filtered noise at various discrete time sample rates can be seen in Figure 17. As expected, the kinematic boundary extends as rates decrease. The instability in the aft quadrant seen in Figure 6 is also dramatically reduced. This may be a result of faster guidance response time. The kinematic boundary appears to have an outer limit regardless of sample rate. This is expected, as the missile is given a finite amount of kinetic energy. The range difference

between the kinematic boundaries is shown in Figure 18 using the baseline as a reference. The smaller rate appears as a positive difference, or improvement in maximum effective range; larger rates are shown as a negative difference, or deterioration. Clearly Figures 17 and 18 support our supposition that lower rates provide better performance.
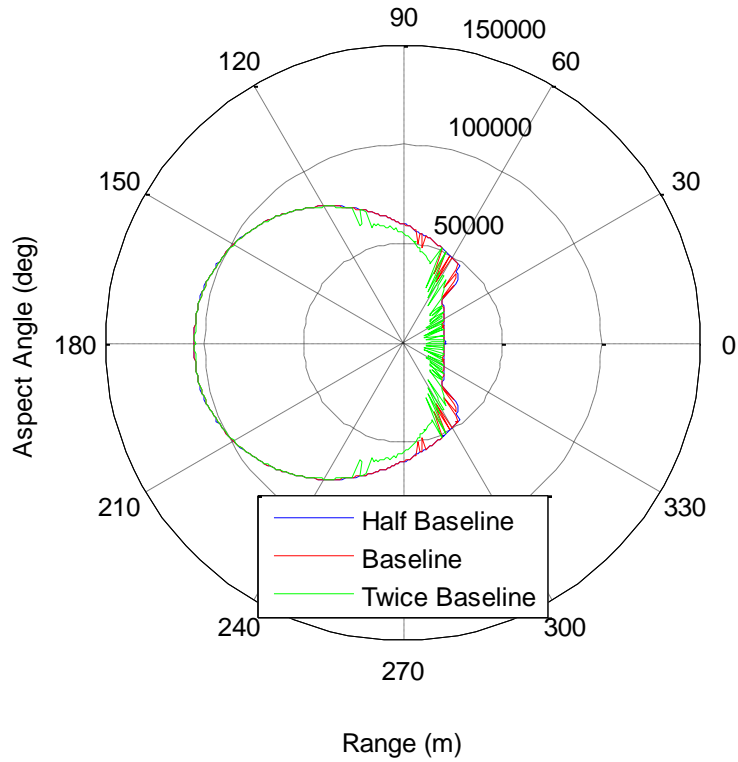
Figure 17.   Kinematic boundaries for PN guidance with Kalman filtered noise at various discrete time step sizes.
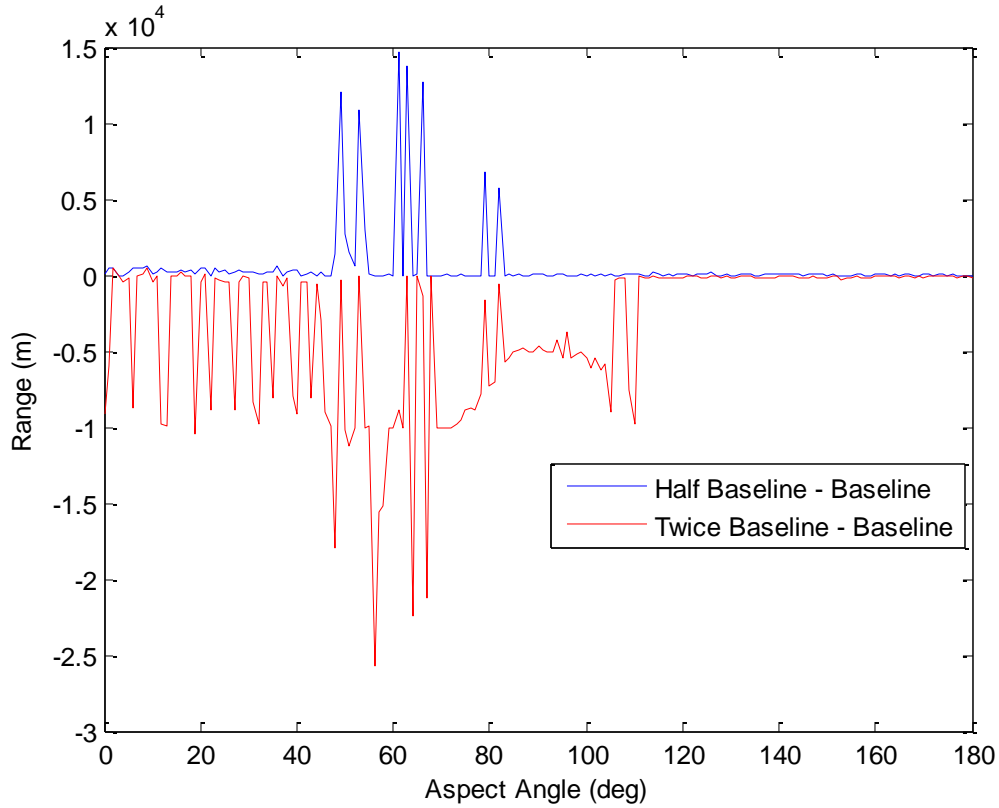
Figure 18.   Difference plot of PN maximum effective range with Kalman filtered
noise at various discrete time step sizes.

Kinematic boundaries for DG guidance law performance with Kalman filtered noise at various discrete time sample rates can be seen in Figure 19. The range difference between the kinematic boundaries is shown in Figure 20 using the baseline as a reference. The effects of sample rate are not as clear as that of PN. In the forward quadrant, the kinematic boundary extends with lower sample intervals as expected. In the aft quadrant, a lower sample interval causes more deterioration and a higher sample interval shows improvement at approximately 45 degrees. This may be a result of a lower sample interval causing more erratic commanded acceleration.
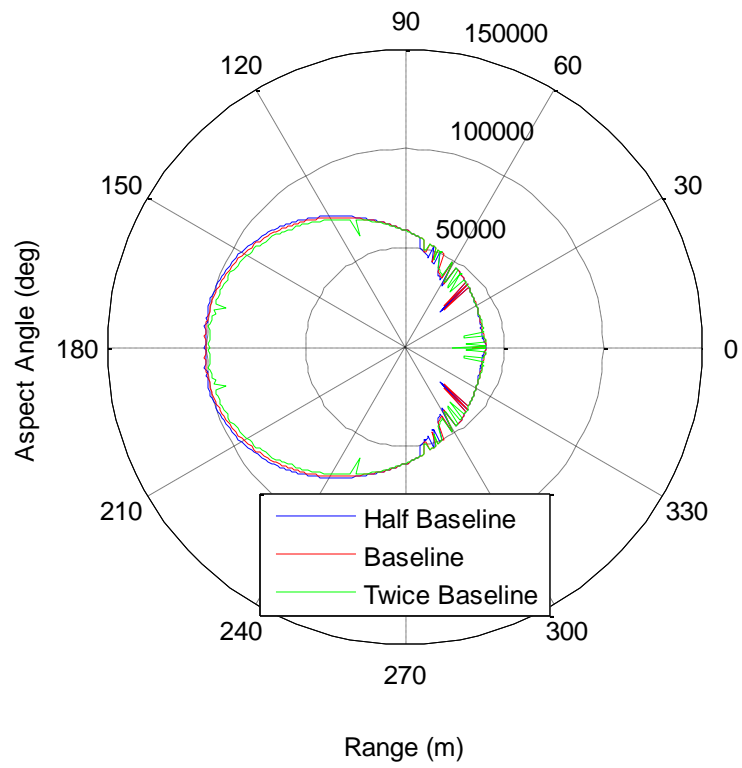
Figure 19. Kinematic boundaries for DG guidance with Kalman filtered noise at various discrete time step sizes.
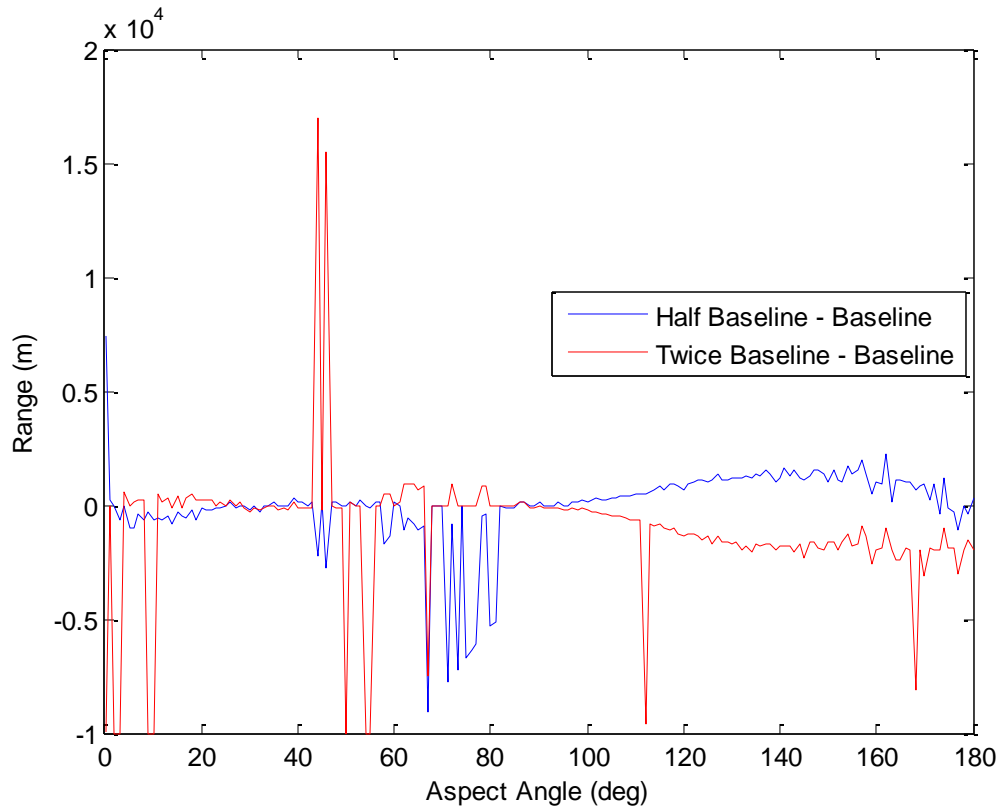
Figure 20.   Difference plot of DG maximum effective range with Kalman filtered
noise at various discrete time step sizes.

In Chapter V, we presented our simulation results in support of our thesis objectives. Chapter VI provides a summary of our conclusions and recommendations for areas of future research.

VI.   CONCLUSIONS AND RECOMMENDATIONS

A.   CONCLUSIONS

The objectives of this research were met. The DG guidance law has been proven to be more sensitive to Kalman filtered noise than PN guidance. While DG guidance suffers slightly larger noise degradation in most aspect angles, it provides a distinct and significant increase in maximum effective range over PN in tail chase scenarios. Based on these results we believe DG does provide a tactical advantage over PN guidance for the simulated AMRAAM.

Analysis of the kinematic boundaries shows instability in the aft quadrant of both PN and DG simulations under noiseless and Kalman filtered noise conditions. The instability was generally reduced as sample interval was lowered, which is expected since missile guidance response time is also lowered. This instability is likely due to transition from supersonic to subsonic speeds. As the missile's speed approaches the transonic region, the parasitic drag coefficient, shown in Figure 5, rises exponentially. If the missile is not maneuvering, transition through the transonic region is accompanied by a slight decrease in missile speed due to parasitic drag and is otherwise uneventful. If the missile is maneuvering, we believe the exponential increase in parasitic drag causes the magnitude of guidance acceleration to also rise proportionately to maintain the desired missile maneuver. Exponential rise in guidance acceleration causes an exponential rise in the induced drag coefficient. This combination of parasitic and induced drag drastically reduces missile speed. Target maneuvers and corresponding missile maneuvers can be expected in the end game of air-to-air missile scenarios. Aspect angles in the aft quadrant correspond to scenarios in which the missile is passing through the transonic region in the end game of the scenario resulting in kinematic boundary instability. This theory is supported by Pehr who experienced regions of instability, which he eliminated by slightly smoothing the parasitic drag coefficient curve in the transonic region [10]. The effects of the transonic region on the kinematic boundary are consistent with noiseless simulations. Since Kalman filtered noise introduces unpredictable accelerations into the model, the kinematic boundary effects vary slightly with each boundary produced. The transonic

instability in DG simulations with Kalman filtered noise may be compounded by the extreme noise sensitivity of DG target state estimates.

The definition of 70 percent efficiency, 14 of 20 simulations, proved to be adequate. Running at least 14 simulations at each tested range, at each angle, to generate a single point on the kinematic boundary is time intensive. Kinematic boundaries using a seven of 10 definition showed significant instability and could not be duplicated easily. Kinematic boundaries using 35 of 50 simulations were estimated to take six to eight days to generate.

Both PN and DG guidance showed more noise sensitivity in the aft quadrant. This is likely due to the higher missile velocity in the end game of simulations with a tail chase trajectory. Target maneuvers, coupled with high missile velocity, produce a larger speed differential for the guidance to overcome.

Our sample rate study showed performance does improve with lower discrete time sample intervals. As onboard processors improve, effective missile range should increase. The results for DG guidance sample rate sensitivity in the transonic region proved puzzling and merits further investigation.

B.   RECOMMENDATIONS FOR FURTHER RESEARCH

In this thesis, the effects of Kalman filtered noise on the performance of PN and DG guidance laws were investigated. Future work in this area may address the following issues.

1.   Instability Investigation

Instability in the aft quadrant is prevalent in all kinematic boundaries in this thesis. The instability was generally reduced as sample interval was lowered. Our sample rate study for DG provides counter-intuitive results as illustrated in Figures 19 and 20. Future work could investigate this instability and our theory for its existence. Possible modifications to the drag model could eliminate this instability completely.

2. Kalman Filter Improvement

Our implementation of Kalman filtering can be modified. We developed estimates of range and LOS angle time derivatives with separate filters. A more advanced filter might incorporate both range and LOS angle measurements into a single filter to estimate the target state directly. This filter would be more complicated but might improve DG guidance performance. We could also modify the filters to incorporate direct range rate measurements from a Doppler type sensor.

3. Classified AMRAAM Guidance Study

The three DOF model in this thesis is based on open source data for the AMRAAM air-to-air missile. A classified thesis incorporating current AMRAAM specifications would make this model more closely parallel the currently fielded version of the missile. Simulations with actual AMRAAM parameters may improve our understanding of the tactical advantage DG guidance provides.

4. Adaptive Guidance Law Switching

The DG guidance law has not presented a clear advantage over PN in all quadrants as illustrated in Figures 12 and 14. An adaptive approach that allows the missile to switch between guidance laws based on the trajectory at launch, or even in flight, may provide superior results.

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX   MATLAB® CODE

The script and function files used in the three DOF simulation are contained in this appendix. The filenames and their purpose are summarized in Table 2.

Table 2.   Summary of MATLAB® files and their purpose.

FILENAME	PURPOSE
A. Simulation run script files	
init.m	Initializes all simulation run script files.
kb_generator.m	Generates a kinematic boundary.
sim_kb.m	Runs a simulation for the kb_generator.m file.
sim_single.m	Runs a single simulation.
max_noise.m	Generates a plot of maximum sustainable noise.
B. Simulation guidance law files	
propnav.m	Implements the proportional navigation guidance law.
diffgeo.m	Implements the differential geometry guidance law.
C. Simulation function files	
time_to_impact.m	Calculates time remaining before impact with the target.
mach_speed.m	Calculates the mach number of missile speed.
rho_value.m	Calculates the density of air at a given altitude.
cdp_value.m	Calculates the parasitic drag coefficient from Figure 5.
fdp_value.m	Calculates the force on the missile due to parasitic drag.
fdi_value.m	Calculates the force on the missile due to induced drag.
noisy_range.m	Adds noise to range measurements.
noisy_theta.m	Adds noise to LOS angle measurements.
missile_motion.m	Updates the missile state vector.
target_motion.m	Updates the target state vector.
D. Simulation filter files	
kalman_dg_range.m	Generates corrected estimates of the range state for the differential geometry guidance law.
kalman_dg_theta.m	Generates corrected estimates of the LOS angle state for the differential geometry guidance law.
kalman_pn_range.m	Generates corrected estimates of the range state for the proportional navigation guidance law.
kalman_pn_theta.m	Generates corrected estimates of the LOS angle state for the proportional navigation guidance law.

A.   SIMULATION RUN SCRIPT FILES

```
% INIT
% Generates global variables in the workspace.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%    File:              init.m
%    Name:              LT Adam Osborn
%    Component Runtime: 8.1 (R2013a)
%    Compiler:          4.18.1 (R2013a)
%                       32-bit (Windows XP)
%    Date:              06 February 2014
%    Description:       Generates global variables in the workspace.
%                       Allows user to modify initialization parameters
%                       for a single simulation using single_sim.m.
%    Inputs:            Provided by the user.
%    Outputs:           Global variables in the workspace.
%    Process:
%    Assumptions:
%    Comments:
%------------------------------------------------------------------------
%------------------------------------------------------------------------


%------ define globals ------
global GRAV DELTA STP_TM ALT INITSPD TRN_TM TRN_G BST_TM PLOTS RUNS...
    DEGSTEP MASS DIAM SREF eAR FBST PN_MAXG DG_MAXG NPRM SIG_THETA...
    SIG_RNG Q2_PN_THETA Q2_PN_RNG Q2_SHIFT Q2_DG_THETA1 Q2_DG_THETA2...
    Q2_DG_RNG INITRNG INITHDG NZ_FACTOR NZ LAW FILT

%------ define input vector ------
INITRNG = 30000;    % Initial range (m) between missile and target.
INITHDG = 10;       % Initial heading (deg) of the target.
NZ_FACTOR = 1;      % Noise factor used as a multiplier to randn noise
                    % added in noisy_theta.m and noisy_range.m.
NZ = 1;             % Adds noise to simulated sensor readings of theta
                    % and range if "ON". Set to 0 for "OFF" and 1 for
                    % "ON".
LAW = 1;            % Selects the guidance law to be implemented. Set
                    % to 1 for PN and 2 for DG.
FILT = 1;           % Guidance law uses kalman filter estimates to
                    % calculate guidance acceleration when set to "ON"
                    % and actual parameters when "OFF". If "OFF"
                    % Kalman filters still run and generate plots but
                    % outputs are not used
                    % for guidance. Set to 0 for "OFF" and 1 for "ON".
PLOTS = 1;          % Allows single_sim.m to plot scenario and filter
                    % information when set to "ON". No plots are
                    % generated when set to "OFF". Set to 0 for "OFF"
                    % and 1 for "ON".
RUNS = 20;          % Number of simulations used to determine missile
                    % effectivness.

%------ define constants ------
% Scenario Constants
GRAV = 9.8045;      % Gravitational constant (m/sec^2).
```

```matlab
DELTA = .01;          % Discrete step size (sec).
STP_TM = 300;         % Maximum scenario run time (sec).
ALT = -6000;          % Constant altitude (m) of missile and target in
                      % NED coordinates given in Pehr thesis pg. 25.
INITSPD = 0.83 * mach_speed(ALT);  % Initial speed of missile and
                                   % target are Mach 0.83 or 262.0310
                                   % (m/sec) given in Pehr thesis
                                   % pg. 25.
TRN_TM = 3;           % Time remaining before impact when target begins
                      % turn given in Pehr thesis pg. 25. Set to 0 for
                      % no turn.
TRN_G = 6;            % Acceleration of target turn given in Pehr thesis
                      % pg. 25.
BST_TM = 6;           % Number of time steps boost is applied.
DEGSTEP = 1;          % Heading increment for kb_generator.m and
                      % max_noise.m simulation files.


% Missile Constants
MASS = 156.8;         % Mass of missile (kg) given in Pehr thesis pg. 74.
DIAM = 0.1778;        % Diameter of missile (m) given in Pehr thesis
                      % pg. 74
SREF = pi*DIAM^2/4;   % Missile cross sectional given in Pehr thesis
                      % pg. 75
eAR = 1.5;            % Unitless elliptical eff & aspect ratio given in
                      % Pehr thesis pg. 88
FBST = 23000;         % Constant thrust (N) during boost given in Pehr
                      % thesis pg. 12
PN_MAXG = 50;         % Maximum PN missile guidance acceleration (g).
DG_MAXG = 15;         % Maximum DG missile guidance acceleration (g).


% Filter Constants
NPRM = 5;             % Guidance gain used by Pehr for all guidance laws
                      % given in Pehr thesis pg. 25.
SIG_THETA = NZ_FACTOR*0.001; % LOS angle sensor uncertainty (rad) given
                             % in Pehr thesis pg. 16.
SIG_RNG = NZ_FACTOR*10;      % Range sensor uncertainty (m) given in
                             % Pehr thesis pg. 16.
Q2_PN_THETA = 1;      % PN theta kalman filter plant noise covariance
                      % multiplier.
Q2_PN_RNG = 1;        % PN range kalman filter plant noise covariance
                      % multiplier.
Q2_SHIFT = 10000;     % Range (m) at which plant noise covariance
                      % multiplier of the DG theta kalman filter shifts
                      % from Q2_DG_THETA1 to Q2_DG_THETA2.
Q2_DG_THETA1 = 1;     % DG theta kalman filter plant noise covariance
                      % multiplier outside Q2_SHIFT range.
Q2_DG_THETA2 = 100;   % DG theta kalman filter plant noise covariance
                      % multiplier inside Q2_SHIFT range.
Q2_DG_RNG = 10;       % DG range kalman filter plant noise covariance
                      % multiplier.


%------ initialize variables ------
%------ functions ------
```

```matlab
function [rho] = kb_generator()
% KB_GENERATOR
% Generates a kinematic boundary.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%   File:               kb_generator.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Runs multiple simulations using kb_sim.m to
%                       generate a kinematic boundary of the maximimum
%                       effective missile range.
%   Inputs:             Global variables provided by user in init.m
%                       file.
%   Outputs:            Kinematic boundary of the maximimum effective
%                       missile range.
%                       Rho vector of maximum ranges. (rho) (m)
%   Process:
%   Assumptions:
%   Comments:           For simulations without noise the missile will
%        `              reach the kill radius from this range in 100%
%                       of simulations. Simulations with noise are
%                       inconsistent near the boundary. To produce a
%                       useful kinematic boundary, the boundary is
%                       marked at the range from which the missile will
%        `              reach the kill radius in 70% of simulations.
%                       Portions of this code have been reused from
%                       Pehr's Thesis.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%------ Initialize MATLAB® Workspace ------
format long;    % Scaled fixed point display format for all float
                % variables with 15 digits for double and 7 digits
                % for single.


close all;      % Closes all previous figures.


clear all;      % Clears all previous workspace variables.


init;           % Runs init.m to load global workspace variables.


%------ define globals ------
global DEGSTEP INITSPD ALT RUNS


%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
tic;    % Start timer for kinematic boundary generation time.


% Initialize current maximum range matrix
rho = zeros(((180/DEGSTEP)+1),1);
```

70

```matlab
%------ functions ------
% Evaluate maximum effective range at each aspect angle
for kk = 1:((180/DEGSTEP)+1)     % One cycle for each aspect angle.

    % Determine target state variables for this aspect angle
    T_hdg = (kk-1)*DEGSTEP; % Set target heading to aspect angle
    x_spd = INITSPD*cos(T_hdg*pi/180); % Component of target speed
                                       % along x-axis.
    y_spd = INITSPD*sin(T_hdg*pi/180); % Component of target speed
                                       % along y-axis.

    load Final_Plotting_Workspace.mat   % Uses ranges from simulations
with RUNS = 10
                        % to speed up production of plots with
                        % RUNS = 20.
    if LAW == 1
            first_rng = PN_noise(kk);
            elseif LAW ==2
                first_rng = DG_noise(kk);
    end
    % First test loop (10km step size)
    for T_rng = first_rng : 10000 : 150000
        disp(['*** ',num2str(T_rng),', 10km step size ***'])

        % Set initial target state
        T_init = [T_rng;x_spd;0;y_spd;ALT;0];

        %Reset variables
        misses = 0;
        hits = 0;
        swings = 0;

        % Determine if missile is effective at this range (100%
        % effective for no noise, 70% effective for noisy simulations)
        while swings <= RUNS

            % Run simulation
            [rngout] = sim_kb(T_init);

            % Determine if missile hit
            if min(rngout)<=5
                disp(['>>> HIT, Heading ',num2str(T_hdg),...
                    ' deg, Range ',num2str(T_rng), ' <<<'])
                hits = hits + 1;         % Count number of hits.
            else
                disp(['>>> MISS, Heading ',num2str(T_hdg),...
                    ' deg, Range ',num2str(T_rng), ' <<<'])
                misses = misses + 1;     % Count number of misses.
            end

            % Missile is ineffective if it misses 30% of the time
            if misses == 0.3*RUNS;
                max_rng = T_rng-10000;  % Save missile's longest
                                        % effective range.
```

71

```matlab
            break
        end

        % Missile is effective if it hits 70% of the time
        if hits == 0.7*RUNS;
            max_rng = T_rng;          % Save missile's longest
                                      % effective range.
            break
        end

        swings = swings + 1;          % Increment simulation count
    end

    % If missile is ineffective, move to the next test loop
    if misses == 0.3*RUNS;
        break
    end
end

% Second test loop (1km step size)
for T_rng = max_rng+1000 : 1000 : max_rng + 9000
    disp(['*** ',num2str(T_rng),', 1km step size ***'])

    % Set initial target state
    T_init = [T_rng;x_spd;0;y_spd;ALT;0];

    %Reset variables
    misses = 0;
    hits = 0;
    swings = 0;

    % Determine if missile is effective at this range (100%
    % effective for no noise, 70% effective for noisy simulations)
    while swings <= RUNS

        % Run simulation
        [rngout] = sim_kb(T_init);

        % Determine if missile hit
        if min(rngout)<=5
            disp(['>>> HIT, Heading ',num2str(T_hdg),...
                ' deg, Range ',num2str(T_rng), ' <<<'])
            hits = hits + 1;          % Count number of hits.
        else
            disp(['>>> MISS, Heading ',num2str(T_hdg),...
                ' deg, Range ',num2str(T_rng), ' <<<'])
            misses = misses + 1;      % Count number of misses.
        end

        % Missile is ineffective if it misses 30% of the time
        if misses == 0.3*RUNS;
            max_rng = T_rng-1000;     % Save missile's longest
                                      % effective range.
```

72

```matlab
            break
        end

        % Missile is effective if it hits 70% of the time
        if hits == 0.7*RUNS;
            max_rng = T_rng;       % Save missile's longest
                                   % effective range.
            break
        end

        swings = swings + 1; % Increment simulation count.
    end

    % If missile is ineffective, move to the next test loop
    if misses == 0.3*RUNS;
        break
    end
end

% Third test loop (100m step size)
for T_rng = max_rng+100 : 100 : max_rng + 900
    disp(['*** ',num2str(T_rng),', 100 m step size ***'])

    % Set initial target state
    T_init = [T_rng;x_spd;0;y_spd;ALT;0];

    %Reset variables
    misses = 0;
    hits = 0;
    swings = 0;

    % Determine if missile is effective at this range (100%
    % effective for no noise, 70% effective for noisy simulations)
    while swings <= RUNS

        % Run simulation
        [rngout] = sim_kb(T_init);

        % Determine if missile hit
        if min(rngout)<=5
            disp(['>>> HIT, Heading ',num2str(T_hdg),...
                ' deg, Range ',num2str(T_rng), ' <<<'])
            hits = hits + 1;       % Count number of hits.
        else
            disp(['>>> MISS, Heading ',num2str(T_hdg),...
                ' deg, Range ',num2str(T_rng), ' <<<'])
            misses = misses + 1;   % Count number of misses.
        end

        % Missile is ineffective if it misses 30% of the time
        if misses == 0.3*RUNS;
            max_rng = T_rng-100;   % Save missile's longest
                                   % effective range.
```

73

```matlab
                break
            end

            % Missile is effective if it hits 70% of the time
            if hits == 0.7*RUNS;
                max_rng = T_rng;         % Save missile's longest
                                         % effective range.
                break
            end

            swings = swings + 1;         % Increment simulation count.
        end

        % If missile is ineffective, move to the next test loop
        if misses == 0.3*RUNS;
            break
        end
    end

    % Update rho plotting vector with maximum effective range at this
    % aspect angle
    rho(kk) = max_rng;

end


toc     % Stop timer for kinematic boundary generation time.

% Generate theta vector for plotting
theta = pi/180*(0:DEGSTEP:180);
theta_plot = [theta, -1*fliplr(theta)]'; % Mirror the 180 degree arc of
                                         % theta to create a full 360
                                         % kinematic boundary.


% Generate rho vector for plotting
rho_plot = [rho; flipud(rho)];           % Mirror the 180 degree arc of
                                         % range to create a full 360
                                         % kinematic boundary.


% Plot kinematic boundary
figure(1)
polar (theta_plot, rho_plot)
```

```matlab
function [rngout] = sim_kb(T_INIT)
% SIM_KB
% Conducts a single simulation for the automated kinematic boundary
% generator in kb.m.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:               sim_kb.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Conducts a single simulation for the automated
%                       kinematic boundary generator in kb_generator.m.
%   Inputs:             Initial target state (T_INIT).
%   Outputs:            Actual range output matrix (rngout) (m)
%   Process:            The initial target state vector is given. The
%                       missile state vector is determined. Actual
%                       measurements of LOS angle and range are
%                       created. Gaussian white noise is added to the
%                       actual measurements of LOS angle and range. The
%                       Kalman filters generate corrected estimates of
%                       LOS angle and range time derivatives. These
%                       estimates are used to generate guidance
%                       acceleration. The missile and target state
%                       vectors are updated and the process is
%                       repeated.
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%------ Initialize MATLAB® Workspace ------
format long;    % Scaled fixed point display format for all float
                % variables with 15 digits for double and 7 digits
                % for single.


%------ define globals ------
global GRAV DELTA STP_TM ALT INITSPD TRN_TM MASS FBST BST_TM ...
    SIG_THETA SIG_RNG INITRNG NZ LAW FILT PN_MAXG DG_MAXG

%------ define constants ------
% Position and velocity matrices.
Hp = [1, 0, 0, 0, 0, 0;   % Extracts position components from the
      0, 0, 1, 0, 0, 0;   % missile or target state vector.
      0, 0, 0, 0, 1, 0];

Hv = [0, 1, 0, 0, 0, 0;   % Extracts velocity components from the
      0, 0, 0, 1, 0, 0;   % missile or target state vector.
      0, 0, 0, 0, 0, 1];

% Missile boost acceleration (g).
mab = FBST/(MASS*GRAV);

%------ define input vector ------
```

75

```
% Initial target state vector: [x; vx; y; vy; z; vz]
T = T_INIT;      % Input vector from kinematic boundary file
                 % kb_generator.m.


% Initial missile state vector: [x; vx; y; vy; z; vz]
M = [0;          % Start missile pointing target at (0,0) and
     INITSPD;    % co-altitude of 6000 meters.
     0;
     0;
     ALT;
     0];


%------ initialize variables ------
% Turn value
turn = 0;    % Initial target motion is straight. Turn value is 0 when
             % target motion is straight, 1 when turning.


% Boost value
boost = 1;   % Simulation begins when "boost phase" is initiated. Boost
             % value is 1 during "boost phase" (first 6 sec) and 0
             % afterward.


% Filter inputs
Fdp = fdp_value(ALT, INITSPD, boost);     % Initial parasitic drag
                                          % force (N).


Mag = [0;    % Guidance is not applied until after
       0;    % filters are initialized.
       0];


Fdi = fdi_value(Mag, INITSPD, ALT);     % Initial induced drag
                                        % force (N).


mad = (Fdp + Fdi)/(MASS*GRAV);          % Magnitude of acceleration
                                        % due to drag (g).


mac_paraL_k1 = mad - mab; % Magnitude of acceleration (g) parallel to
                          % the LOS. Signs are in terms of the effect
                          % on the LOS. The acceleration due to drag
                          % (mad) is positive because it opens range.
                          % The acceleration due to boost (mab) is
                          % negative because it closes range.The entire
                          % magnitude of mad and mab are parallel to
                          % the LOS at intialization because the
                          % missile points the target.


mac_perpL_k1 = 0;         % Magnitude of acceleration (g) parallel to
                          % the LOS is zero at initialization since all
                          % missile velocity is pointing at the target.


Ma = [-mac_paraL_k1; 0; 0;];    % Total missile acceleration at
                                % initialization is all in LOS.
                                % mac_paraL_k1 is negative since it is
```

```matlab
                                % in terms of the effect on the LOS
                                % while Ma is in terms of the effect on
                                % the missile.

thetaLM_k0 = 0;   % The DG guidance law uses the previous time step
                  % corrected estimate of thetaLM (thetaLM_k0) when the
                  % guidance law is receiving kalman filter corrected
                  % estimates vice actual parameters (FILT = 1).
                  % Initialized at zero because the missile is pointing
                  % at the target.

thetaLM_k1 = 0;   % The DG and PN guidance laws uses the current time
                  % step corrected estimate of thetaLM (thetaLM_k1) when
                  % the guidance law is receiving kalman filter corrected
                  % estimates vice actual parameters (FILT = 1). The
                  % kalman filters are not initialized until the fourth
                  % time step (kk = 4). Setting thetaLM_k1 to zero for
                  % the first four time steps has no effect since the
                  % output of the guidance laws are also set to zero
                  % until kk = 4 and kk = BST_TM/DELTA for PN and DG
                  % respectively.

VM_old = Hv*M;    % The DG guidance law uses the missile velocity vector
                  % (VM) from the previous time step (VM_old) when the
                  % guidance law is receiving kalman filter corrected
                  % estimates vice actual parameters (FILT = 1).
                  % Initialized to actual VM.

thetadt_old = T(4)/INITRNG; % Initialized to the actual LOS angle rate.
                            % Used for calculating actual LOS angle
                            % acceleration (thetadtdt).

rngdt_old = 0;    % Initialized to the actual range rate. Used for
                  % calculating actual range acceleration (rangedtdt).

% Output matrices (Built in advance to speed MATLAB® processing.)
% Missile guidance acceleration (g).
magout = zeros(1, (STP_TM/DELTA));
% Actual LOS angle (rad).
thetaout = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of LOS angle (rad).
theta_k1out = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of LOS angle rate (rad/sec).
thetadt_k1out = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of LOS angle acceleration
% (rad/sec^2).
thetadtdt_k1out = zeros(1, (STP_TM/DELTA));

% Actual range to target (m) output matrix.
rngout = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of range to target (m).
rng_k1out = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of range rate to target (m/sec).
rngdt_k1out = zeros(1, (STP_TM/DELTA));
```

```matlab
% Current time step corrected estimate of range acceleration to target
% (m/sec^2).
rngdtdt_k1out = zeros(1, (STP_TM/DELTA));


%------ functions ------
% Primary Loop
for kk = 1:(STP_TM/DELTA)

    % Calculate actual scenario parameters
    malt = M(5);        % Missile altitude (m) in NED coordinates
    VM = Hv*M;          % Missile velocity vector
    VMu = VM/norm(VM);  % Unit vector in direction of VM
    mspd = norm(VM);    % Missile speed (m/sec)
    VT = Hv*T;          % Target velocity vector
    tspd = norm(VT);    % Target speed (m/sec)
    PL = Hp*(T - M);    % LOS position vector from missile to target
    PLu = PL/norm(PL);  % Unit vector in the direction of PLu
    VL = Hv*(T - M);    % LOS velocity vector from missile to target
    rng = norm(PL);     % Range (m)
    rngdt = VL'*PLu;    % Range rate (m/sec)
    theta = atan2(PL(2), PL(1));        % LOS angle (rad)
    thetaVM = atan2(VM(2), VM(1));      % Angle between missile velocity
                                        % vector and x-axis.
    thetaLM = thetaVM - theta;          % Missile lead angle defined as
                                        % the angle between PL and VM.


    % Generate noisy measurements
    if NZ == 0          % Adds noise to simulated sensor readings of
                        % theta and range if "ON". Set to 0 for "OFF"
                        % and 1 for "ON".
        nz_theta = theta;
        nz_rng = rng;
    else                % Noise "ON"
        nz_theta = noisy_theta(theta);
        nz_rng = noisy_range(rng);
    end

    % Generate actual LOS angle rate
    VLperp = VL - rngdt * PLu;       % Portion of LOS velocity vector
                                     % (VL) perpendicular to LOS
                                     % position unit vector (PLu)

    if norm(VLperp) == 0             % Prevent divide by zero.
        VLperpu = cross([0;0;1], PLu);
    else
        VLperpu = VLperp/norm(VLperp);  % Unit vector in direction
                                        % of VLperp.
    end

    sign_thetadt = sign(cross(PLu, VLperpu));   % Sign of LOS angle
                                                % rate depends on the
                                                % direction of VLperpu.
    thetadt = sign_thetadt(3)*norm(VLperp)/rng; % Actual LOS angle rate
```

78

```matlab
% Generate Actual LOS angle acceleration (thetadtdt) and range
% acceleration (rngdtdt)
thetadtdt = (thetadt - thetadt_old)/DELTA;
rngdtdt = (rngdt - rngdt_old)/DELTA;

% Initialize filters using first three time steps
if kk == 4
    % Previous time step corrected estimate of LOS angle (rad).
    theta_k0 = thetaout(1,3);
    % Previous time step corrected estimate of LOS angle rate
    % (rad/sec).
    thetadt_k0 = (thetaout(1,3) - thetaout(1,2))/DELTA;
    % Previous time step corrected estimate of LOS angle
    % acceleration(rad/sec^2).
    thetadtdt_k0 = (((thetaout(1,3) - thetaout(1,2))/DELTA)-...
        ((thetaout(1,2) - thetaout(1,1))/DELTA))/DELTA;

    % Previous time step corrected estimate of range (m).
    rng_k0 = rngout(1,3);
    % Previous time step corrected estimate of range rate (m/sec).
    rngdt_k0 = (rngout(1,3) - rngout(1,2))/DELTA;
    % Previous time step corrected estimate of range acceleration
    % (m/sec^2).
    rngdtdt_k0 =(((rngout(1,3) - rngout(1,2))/DELTA)-...
        ((rngout(1,2) - rngout(1,1))/DELTA))/DELTA;

    % Initialize state covariance matrices
    if LAW == 1 % PN
        % Initialized with simulated LOS angle sensor accuracy.
        Ptheta_old = diag([SIG_THETA^2;
                          (2/(DELTA^2))*SIG_THETA^2]);
        % Initialized with simulated range sensor accuracy.
        Prng_old = diag([SIG_RNG^2;
                        (2/(DELTA^2))*SIG_RNG^2]);

    elseif LAW == 2 % DG
        % Initialized with simulated LOS angle sensor accuracy.
        Ptheta_old = diag([SIG_THETA^2;
                          (2/(DELTA^2))*SIG_THETA^2;
                          (4/(DELTA^4))*SIG_THETA^2]);
        % Initialized with simulated range sensor accuracy.
        Prng_old = diag([SIG_RNG^2;
                        (2/(DELTA^2))*SIG_RNG^2;
                        (4/(DELTA^4))*SIG_RNG^2]);
    end
end

% Kalman filters
if kk >= 4
    if LAW == 1 %PN filters
        % Current time step corrected estimate of range and its
        % time derivatives.
        [ rng_k1, rngdt_k1, Prng ] = kalman_pn_range(rng_k0,...
            rngdt_k0, nz_rng, Prng_old, mac_paraL_k1);
```

```matlab
            % Current time step corrected estimate of LOS angle and its
            % time derivatives.
            [ theta_k1, thetadt_k1, Ptheta ] = kalman_pn_theta...
                (theta_k0, thetadt_k0, nz_theta, Ptheta_old, ...
                mac_perpL_k1, rng_k1);

        elseif LAW == 2 %DG filters
            % Current time step corrected estimate of range and its
            % time derivatives.
            [ rng_k1, rngdt_k1, rngdtdt_k1, Prng ] = kalman_dg_range...
                (rng_k0, rngdt_k0, rngdtdt_k0, nz_rng, Prng_old, ...
                mac_paraL_k1);

            % Current time step corrected estimate of LOS angle and its
            % time derivatives.
            [ theta_k1, thetadt_k1, thetadtdt_k1, Ptheta ] = ...
                kalman_dg_theta(theta_k0, thetadt_k0, thetadtdt_k0, ...
                nz_theta, Ptheta_old, mac_perpL_k1, rng_k1);

            % Adjust corrected estimates to include deterministic
            % acceleration inputs.
            thetadtdtplot_k1 = thetadtdt_k1 +
(mac_perpL_k1*GRAV/rng_k1);
                        % Update current time step corrected estimate
                        % of LOS angle acceleration (thetadtdt_k1)to
                        % include the deterministic acceleration input
                        % (missile's drag and boost accelerations
                        % perpendicular to the LOS). This term cannot
                        % be added inside the filter because changes
                        % in LOS angle acceleration are modeled by the
                        % filter as white noise.

            rngdtdtplot_k1 = rngdtdt_k1 + (mac_paraL_k1*GRAV);
                        % Update current time step corrected estimate
                        % of range acceleration (rangedtdt_k1) to
                        % include the deterministic acceleration input
                        % (missile's drag and boost accelerations
                        % parallel to the LOS). This term cannot be
                        % added inside the filter because changes in
                        % range acceleration are modeled by the filter
                        % as white noise.
        end

        % Generate current time step corrected estimate of missile
        % lead angle (thetaLM_k1) for guidance law.
        thetaLM_k1 = thetaVM - theta_k1;
                        % Corrected estimate of angle between LOS and
                        % VM for accel perpendicular to VM
    end

    % Missile guidance
    if LAW == 1          % PN guidance.
```

```matlab
    if FILT == 0     % Actual scenario parameters given to guidance
                     % law in propnav.m.
        [ mag ] = propnav(thetaLM, thetadt, rngdt);
                     % PN guidance acceleration (g).

    else             % Kalman filter estimates given to guidance
                     % law in propnav.m.

        if kk < 4    % Delays guidance acceleration until after
                     % Kalman filters have been initialized.
            mag = 0;% Sets guidance acceleration to zero.

        else
            [ mag ] = propnav(thetaLM_k1, thetadt_k1, rngdt_k1);
                     % PN guidance acceleration (g).
        end
    end
    % Limit guidance acceleration to MAXG to prevent excessive
    % drag and missile damage.
    if abs(mag) > PN_MAXG
        mag = sign(mag)*PN_MAXG; % Reassigns guidance acceleration
                                 % to the maximum value while still
                                 % retaining the sign.
    end
end

if LAW == 2          % DG guidance.

    if FILT == 0     % Actual scenario parameters given to guidance
                     % law in diffgeo.m. The previous time step
                     % missile acceleration vector (Ma) is used
                     % since the output of this function affects the
                     % current time step Ma. Using the previous time
                     % step Ma introduces some error into the
                     % guidance but since the filter is "OFF"
                     % (FILT = 0) the noise switch must also be
                     % "OFF" (NZ = 0). The guidance doesn't track
                     % with noise and no filter. With no noise, Ma
                     % is very smooth and the difference between Ma
                     % in one time step is rather small.

        [ mag ] = diffgeo(theta, thetadt, thetadtdt, ...
            rng, rngdt, rngdtdt, VM, Ma, thetaLM);
                     % DG guidance acceleration (g).

    else             % Kalman filter estimates given to guidance law
                     % in diffgeo.m. The previous time step missile
                     % acceleration vector (Ma) is used since the
                     % output of this function affects the current
                     % time step Ma. Using the previous time step Ma
                     % introduces some error into the guidance. With
                     % noise, Ma can be very rough and the
                     % difference between Ma in one time step can be
                     % large. To combat the error introduced, we
```

```matlab
                          % have given the guidance law Kalman filter
                          % estimates from the previous time step so all
                          % information is from the same time step. This
                          % generates a guidance acceleration that is
                          % much more accurate but applied one time step
                          % late. The result is a much smoother guidance
                          % output.

            if kk <= BST_TM/DELTA % Estimates are rough until the
                                  % Kalman filters have a some time to
                                  % settle. Delaying guidance
                                  % acceleration until after boost
                                  % results in much smoother guidance
                                  % output.

                mag = 0; % Sets guidance acceleration to zero.

            else
                [ mag ] = diffgeo(theta_k1out(kk-1),...
                    thetadt_k1out(kk-1), thetadtdt_k1out(kk-1), ...
                    rng_k1out(kk-1), rngdt_k1out(kk-1), ...
                    rngdtdt_k1out(kk-1), VM_old, Ma, thetaLM_k0);
                        % DG guidance acceleration (g).

                % Average guidance output
                mag = (mag + magout(kk-1) + magout(kk-2) ...
                    + magout(kk-3))/4;
                        % Averages guidance output with three previous
                        % outputs. The result is a much smoother
                        % guidance output.
            end
        end
        if abs(mag) > DG_MAXG
            mag = sign(mag)*DG_MAXG; % Reassigns guidance acceleration
                                     % to the maximum value while still
                                     % retaining the sign.
        end
end

% Generate guidance vector
Magu = cross([0;0;1],VMu);
                        % Guidance unit vector (Magu) is perpendicular
                        % to the missile velocity unit vector (VMu).
                        % It is always in x-y plane since both target
                        % and missile remain in x-y plane. The sign of
                        % guidance acceleration controls the direction
                        % of Magu.

Mag = mag*Magu;         % Generates a guidance vector of appropriate
                        % length and direction.

% Missile boost
if kk < (BST_TM/DELTA)  % Boost phase is active.
    Mab = mab*VMu;      % Missile acceleration (g) due to boost.
```

```matlab
                            % Boost thrust is always applied in the
                            % direction of the missile velocity unit
                            % vector (VMu).
    boost = 1;              % Maintains boost phase.
else                       % Boost phase terminated.
    Mab = [0; 0; 0];       % Boost no longer contributes to missile
                            % acceleration.
    boost = 0;             % Terminates boost phase.
end


% Missile Drag
% Determine current parasitic drag coefficient.
Fdp = fdp_value(malt, mspd, boost); % Parasitic drag force (N).

% Determine current induced drag coefficient.
Fdi = fdi_value(Mag, mspd, malt); % Induced drag force (N).


% Determine acceleration due to total drag (g).
mad = (Fdp + Fdi)/(MASS*GRAV);
                    % Magnitude of acceleration due to drag (mad) is
                    % due to parasitic and induced drag.
Mad = mad*(-VMu); % Acceleration due to drag is always applied in
                    % the opposite direction of the missile velocity
                    % unit vector (VMu).


% Calculate deterministic inputs for Kalman filters from
% LOS perpective
if kk >= 4  % Estimated theta and range are not available for
            % deterministic inputs until after the Kalman filters
            % are initialized.
    PL_k1 = [cos(theta_k1)*rng_k1; sin(theta_k1)*rng_k1; 0];
            % Kalman filters only have access to an estimate of the
            % LOS position vector (PL).
    PLu_k1 = PL_k1/norm(PL_k1);    % Estimated LOS position
                                    % unit vector.

    % Combined deterministic missile accelerations (g)
    % due to boost and drag.
    Mac = Mab + Mad;

    % Estimated magnitude of combined deterministic missile
    % acceleration parallel to the LOS (mac_paraL_k1) (g).
    mac_paraL_k1 = Mac'*PLu_k1;

    % Estsimated vector of combined deterministic missile
    % acceleration parallel to the LOS (Mac_paraL_k1) (g).
    Mac_paraL_k1 = mac_paraL_k1*PLu_k1;

    % Estsimated vector of combined deterministic missile
    % acceleration perpendicular to the LOS (Mac_perpL_k1) (g).
    Mac_perpL_k1 = Mac - Mac_paraL_k1;

    % Estimated magnitude of combined deterministic missile
    % acceleration perpendicular to the LOS (mac_perpL_k1) (g).
```

83

```matlab
        mac_perpL_k1 = norm(Mac_perpL_k1);

        % Sign of mac_perpL_k1 gives direction and must be preserved.
        sign_mac_perpL_k1 = ...
            sign(cross((Mac_perpL_k1/norm(Mac_perpL_k1)), PLu));

        % A positive mac_paraL_k1 from the missile's perspective
        % actually causes the LOS range to decrease.
        mac_paraL_k1 = -mac_paraL_k1;

        % Deterministic input mac_perpL_k1 causes theta to increase.
        if sign_mac_perpL_k1(3) < 0

        % A positive mac_perpL_k1 from the missile's perspective
        % actually causes the LOS angle to decrease.
            mac_perpL_k1 = -mac_perpL_k1;
        end
end


% Total missile acceleration
Ma = Mag + Mad + Mab; % Total missile accel in (g) due to Guidance
                      %  (Mag), Drag (Mad), and Boost (Mab).


% Update output matrices
% Missile guidance acceleration (g).
magout(1,kk) = mag;
% Actual LOS angle (rad).
thetaout(1,kk) = theta;
% Actual range to target (m).
rngout(1,kk) = rng;

if kk >= 4  % Available to update only after Kalman filters
            %  have been initialized.
    % Current time step corrected estimate of LOS angle (rad).
    theta_k1out(1,kk) = theta_k1;
    % Current time step corrected estimate of LOS angle
    % rate (rad/sec).
    thetadt_k1out(1,kk) = thetadt_k1;
    % Current time step corrected estimate of range to target (m).
    rng_k1out(1,kk) = rng_k1;
    % Current time step corrected estimate of range rate to
    % target (m/sec).
    rngdt_k1out(1,kk) = rngdt_k1;

    if LAW == 2 % Available only if DG guidance is used.
        % Current time step corrected estimate of LOS angle
        % acceleration (rad/sec^2).
        thetadtdt_k1out(1,kk) = thetadtdtplot_k1;
        % Current time step corrected estimate of range
        % acceleration to target (m/sec^2).
        rngdtdt_k1out(1,kk) = rngdtdtplot_k1;
    end
end
```

```matlab
% Stop criteria
% Actual range rate (rngdt) is opening after boost has
% been terminated.
if (rngdt > 0) && (kk > BST_TM/DELTA)
            % Time delay allows for boost to overcome drag and
            % accelerate missile speed above target speed.
    break   % Stops simulation.
end

% 2)Missile speed is less than target speed
if (mspd < tspd) && (kk > BST_TM/DELTA)
            % Time delay allows for boost to overcome drag and
            % accelerate missile speed above target speed.
    break   % Stops simulation.
end

% 3) Range to target is less than 5m
if (rng < 5)
    break% Stops simulation.
end

% Update filter variables
thetadt_old = thetadt;  % Actual LOS angle rate (rad/sec).
rngdt_old = rngdt;      % Actual range rate (m/sec).
VM_old = VM;            % Actual missile velocity vector.
thetaLM_k0 = thetaLM_k1;% Estimate of angle (rad) between missile
                        % velocity vector and x-axis.

if kk >= 4  % Available to update only after Kalman filters
            % have been initialized.
    theta_k0 = theta_k1;    % Estimate of LOS angle (rad).
    thetadt_k0 = thetadt_k1;% Estimate of LOS angle rate (rad/sec).
    Ptheta_old = Ptheta;    % Theta corrected estimate covariance.

    rng_k0 = rng_k1;        % Estimate of range (m).
    rngdt_k0 = rngdt_k1;    % Estimate of range rate (m/sec).
    Prng_old = Prng;        % Range corrected estimate covariance.

    if LAW == 2 % Available only if DG guidance is used.
        thetadtdt_k0 = thetadtdt_k1;% Estimate of LOS angle
                                    % acceleration (rad/sec^2).
        rngdtdt_k0 = rngdtdt_k1;    % Estimate of range
                                    % acceleration (m/sec^2)
    end
end

% Missile Motion
M = missile_motion(M, Ma);  % Returns updated missile state vector.

% Target motion
if time_to_impact(rng, rngdt) < TRN_TM  % Initiates turn when
                                        % impact is TURN_TIME
                                        % seconds away
    turn = 1;                   % Turn value is 0 when target motion
```

85

```matlab
                                    % is straight, 1 when turning.
    end

    T = target_motion( T, turn );% Returns updated target state vector.
end % End primary loop

% Update outputs to the kinematic boundary file kb_generator.m
% by removing unused columns.
rngout = rngout(1,1:kk);          % Actual range to target (m).
end
```

```matlab
% SIM_SINGLE
% Conducts a single simulation from user inputs in init.m.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%    File:               sim_single.m
%    Name:               LT Adam Osborn
%    Component Runtime:  8.1 (R2013a)
%    Compiler:           4.18.1 (R2013a)
%                        32-bit (Windows XP)
%    Date:               06 February 2014
%    Description:        Conducts a single simulation based on the user
%                        inputs from init.m.
%    Inputs:             Global variables provided by user in init.m
%                        file.
%    Outputs:            Figure (1) - Overall summary of the simulation.
%                        Figure (2) - Summary of actual and estimated
%                        time derivatives of theta.
%                        Figure (3) - Summary of actual and estimated
%                        time derivatives of range.

%    Process:            Inputs are drawn from the init.m file. Target
%                        and missile states are generated. Actual
%                        measurements of LOS angle and range are
%                        created. Gaussian white noise is added to the
%                        actual measurements of LOS angle and range.
%                        The Kalman filters generate corrected estimates
%                        of LOS angle and range time derivatives. These
%                        estimates are used to generate guidance
%                        acceleration. The missile and target state
%                        vectors are updated and the process is
%                        repeated.
%    Assumptions:
%    Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%------ Initialize MATLAB® Workspace ------
format long;      % Scaled fixed point display format for all float
                  % variables with 15 digits for double and 7 digits
                  % for single.


close all;      % Closes all previous figures.


clear all;      % Clears all previous workspace variables.


init;           % Runs init.m to load global workspace variables.


%------ define globals ------
global GRAV DELTA STP_TM ALT INITSPD TRN_TM MASS FBST PN_MAXG...
    DG_MAXG BST_TM SIG_THETA SIG_RNG INITRNG INITHDG NZ LAW FILT PLOTS


%------ define constants ------
% Position and velocity matrices.
Hp = [1, 0, 0, 0, 0, 0;      % Extracts position components from the
      0, 0, 1, 0, 0, 0;      % missile or target state vector.
```

```
      0, 0, 0, 0, 1, 0];

Hv = [0, 1, 0, 0, 0, 0;      % Extracts velocity components from the
      0, 0, 0, 1, 0, 0;      % missile or target state vector.
      0, 0, 0, 0, 0, 1];

% Missile boost acceleration (g).
mab = FBST/(MASS*GRAV);


%------ define input vector ------
% Initial target state vector: [x; vx; y; vy; z; vz]
T = [INITRNG;
     INITSPD*cos(INITHDG*pi/180);
     0;
     INITSPD*sin(INITHDG*pi/180);
     ALT;
     0];

% Initial missile state vector: [x; vx; y; vy; z; vz]
M = [0;          % Start missile pointing target at (0,0) and
     INITSPD;    % co-altitude of 6000 meters.
     0;
     0;
     ALT;
     0];


%------ initialize variables ------
% Turn value
turn = 0;        % Initial target motion is straight. Turn value is 0
                 % when target motion is straight, 1 when turning.


% Boost value
boost = 1;       % Simulation begins when "boost phase" is initiated.
                 % Boost value is 1 during "boost phase" (first 6 sec)
                 % and 0 afterward.


% Filter inputs
Fdp = fdp_value(ALT, INITSPD, boost);   % Initial parasitic drag
                                        % force (N).


Mag = [0;   % Guidance is not applied until after
       0;   % filters are initialized.
       0];


Fdi = fdi_value(Mag, INITSPD, ALT);     % Initial induced drag
                                        % force (N).


mad = (Fdp + Fdi)/(MASS*GRAV);          % Magnitude of acceleration
                                        % due to drag (g).


mac_paraL_k1 = mad - mab; % Magnitude of acceleration (g) parallel to
                          % the LOS. Signs are in terms of the effect
                          % on the LOS. The acceleration due to drag
```

```matlab
                        % (mad) is positive because it opens range.
                        % The acceleration due to boost (mab) is
                        % negative because it closes range.The entire
                        % magnitude of mad and mab are parallel to
                        % the LOS at intialization because the
                        % missile points the target.

mac_perpL_k1 = 0;          % Magnitude of acceleration (g) parallel to
                        % the LOS is zero at initialization since all
                        % missile velocity is pointing at the target.

Ma = [-mac_paraL_k1; 0; 0;];    % Total missile acceleration at
                        % initialization is all in LOS.
                        % mac_paraL_k1 is negative since it is
                        % in terms of the effect on the LOS
                        % while Ma is in terms of the effect on
                        % the missile.

thetaLM_k0 = 0; % The DG guidance law uses the previous time step
                % corrected estimate of thetaLM (thetaLM_k0) when the
                % guidance law is receiving kalman filter corrected
                % estimates vice actual parameters (FILT = 1).
                % Initialized at zero because the missile is pointing
                % at the target.

thetaLM_k1 = 0; % The DG and PN guidance laws uses the current time
                % step corrected estimate of thetaLM (thetaLM_k1) when
                % the guidance law is receiving kalman filter corrected
                % estimates vice actual parameters (FILT = 1). The
                % kalman filters are not initialized until the fourth
                % time step (kk = 4). Setting thetaLM_k1 to zero for
                % the first four time steps has no effect since the
                % output of the guidance laws are also set to zero
                % until kk = 4 and kk = BST_TM/DELTA for PN and DG
                % respectively.

VM_old = Hv*M;  % The DG guidance law uses the missile velocity vector
                % (VM) from the previous time step (VM_old) when the
                % guidance law is receiving kalman filter corrected
                % estimates vice actual parameters (FILT = 1).
                % Initialized to actual VM.

thetadt_old = T(4)/INITRNG; % Initialized to the actual LOS angle rate.
                            % Used for calculating actual LOS angle
                            % acceleration (thetadtdt).

rngdt_old = 0;  % Initialized to the actual range rate. Used for
                % calculating actual range acceleration (rangedtdt).

% Output matrices (Built in advance to speed MATLAB® processing.)
% Target position (m).
Tpout = zeros(1, (STP_TM/DELTA));
% Missile position (m).
Mpout = zeros(1, (STP_TM/DELTA));
```

```matlab
% Missile guidance acceleration (g).
magout = zeros(1, (STP_TM/DELTA));
% Missile drag acceleration (g).
madout = zeros(1, (STP_TM/DELTA));
% Missile speed (m/sec^2).
mspdout = zeros(1, (STP_TM/DELTA));
% Time (sec).
tout = zeros(1, (STP_TM/DELTA));

% Actual LOS angle (rad).
thetaout = zeros(1, (STP_TM/DELTA));
% Noisy LOS angle (rad).
nz_thetaout = zeros(1, (STP_TM/DELTA));
% Actual LOS angle rate (rad/sec).
thetadtout = zeros(1, (STP_TM/DELTA));
% Actual LOS angle acceleration (rad/sec^2).
thetadtdtout = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of LOS angle (rad).
theta_k1out = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of LOS angle rate (rad/sec).
thetadt_k1out = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of LOS angle
% acceleration (rad/sec^2).
thetadtdt_k1out = zeros(1, (STP_TM/DELTA));
% Output matrix for sqrt of maximum eigenvalue of current time step
% theta corrected estimate covariance.
sqePthetaout = zeros(1, (STP_TM/DELTA));

% Actual range to target (m).
rngout = zeros(1, (STP_TM/DELTA));
% Noisy range to target (m).
nz_rngout = zeros(1, (STP_TM/DELTA));
% Actual range rate to target (m/sec).
rngdtout = zeros(1, (STP_TM/DELTA));
% Actual range acceleration to target (m/sec^2).
rngdtdtout = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of range to target (m).
rng_k1out = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of range rate to target (m/sec).
rngdt_k1out = zeros(1, (STP_TM/DELTA));
% Current time step corrected estimate of range acceleration to
% target (m/sec^2).
rngdtdt_k1out = zeros(1, (STP_TM/DELTA));
% Output matrix for sqrt of maximum eigenvalue of current time step
% range corrected estimate covariance.
sqePrngout = zeros(1, (STP_TM/DELTA));

%------ functions ------
% Primary Loop
for kk = 1:(STP_TM/DELTA)

    % Calculate actual scenario parameters
    malt = M(5);         % Missile altitude (m) in NED coordinates
    VM = Hv*M;           % Missile velocity vector
    VMu = VM/norm(VM);   % Unit vector in direction of VM
```

```matlab
mspd = norm(VM);      % Missile speed (m/sec)
VT = Hv*T;            % Target velocity vector
VTu = VT/norm(VT);    % Unit vector in direction of VT
tspd = norm(VT);      % Target speed (m/sec)
PL = Hp*(T - M);      % LOS position vector from missile to target
PLu = PL/norm(PL);    % Unit vector in the direction of PLu
VL = Hv*(T - M);      % LOS velocity vector from missile to target
rng = norm(PL);       % Range (m)
rngdt = VL'*PLu;      % Range rate (m/sec)
theta = atan2(PL(2), PL(1));     % LOS angle (rad)
thetaVM = atan2(VM(2), VM(1));   % Angle between missile velocity
                                 % vector and x-axis.
thetaLM = thetaVM - theta;       % Missile lead angle defined as the
                                 % angle between PL and VM.


% Generate noisy measurements
if NZ == 0            % Adds noise to simulated sensor readings of
                      % theta and range if "ON". Set to 0 for "OFF"
                      % and 1 for "ON".
    nz_theta = theta;
    nz_rng = rng;
else                  % Noise "ON"
    nz_theta = noisy_theta(theta);
    nz_rng = noisy_range(rng);
end


% Generate actual LOS angle rate
VLperp = VL - rngdt * PLu;  % Portion of LOS velocity vector (VL)
                            % perpendicular to LOS position unit
                            % vector (PLu)


if norm(VLperp) == 0        % Prevent divide by zero.
    VLperpu = cross([0;0;1], PLu);
else
    VLperpu = VLperp/norm(VLperp);  % Unit vector in direction
                                    % of VLperp.
end


sign_thetadt = sign(cross(PLu, VLperpu));   % Sign of LOS angle
                                            % rate depends on the
                                            % direction of VLperpu.
thetadt = sign_thetadt(3)*norm(VLperp)/rng; % Actual LOS angle rate

% Generate Actual LOS angle acceleration (thetadtdt) and range
% acceleration (rngdtdt)
thetadtdt = (thetadt - thetadt_old)/DELTA;
rngdtdt = (rngdt - rngdt_old)/DELTA;


% Initialize filters using first three time steps
% Initialize filters using first three time steps
if kk == 4
    % Previous time step corrected estimate of LOS angle (rad).
    theta_k0 = thetaout(1,3);
    % Previous time step corrected estimate of LOS angle rate
```

91

```matlab
    % (rad/sec).
    thetadt_k0 = (thetaout(1,3) - thetaout(1,2))/DELTA;
    % Previous time step corrected estimate of LOS angle
    % acceleration(rad/sec^2).
    thetadtdt_k0 = (((thetaout(1,3) - thetaout(1,2))/DELTA)-...
        ((thetaout(1,2) - thetaout(1,1))/DELTA))/DELTA;

    % Previous time step corrected estimate of range (m).
    rng_k0 = rngout(1,3);
    % Previous time step corrected estimate of range rate (m/sec).
    rngdt_k0 = (rngout(1,3) - rngout(1,2))/DELTA;
    % Previous time step corrected estimate of range acceleration
    % (m/sec^2).
    rngdtdt_k0 =(((rngout(1,3) - rngout(1,2))/DELTA)-...
        ((rngout(1,2) - rngout(1,1))/DELTA))/DELTA;

    if LAW == 1 % PN
        % Initialized with simulated LOS angle sensor accuracy.
        Ptheta_old = diag([SIG_THETA^2;
                            (2/(DELTA^2))*SIG_THETA^2]);
        % Initialized with simulated range sensor accuracy.
        Prng_old = diag([SIG_RNG^2;
                            (2/(DELTA^2))*SIG_RNG^2]);

    elseif LAW == 2 % DG
        % Initialized with simulated LOS angle sensor accuracy.
        Ptheta_old = diag([SIG_THETA^2;
                            (2/(DELTA^2))*SIG_THETA^2;
                            (4/(DELTA^4))*SIG_THETA^2]);
        % Initialized with simulated range sensor accuracy.
        Prng_old = diag([SIG_RNG^2;
                            (2/(DELTA^2))*SIG_RNG^2;
                            (4/(DELTA^4))*SIG_RNG^2]);
    end
end

% Kalman filters
if kk >= 4
    if LAW == 1 %PN filters
        % Current time step corrected estimate of range and its
        % time derivatives.
        [ rng_k1, rngdt_k1, Prng ] = kalman_pn_range(rng_k0,...
            rngdt_k0, nz_rng, Prng_old, mac_paraL_k1);

        % Current time step corrected estimate of LOS angle and its
        % time derivatives.
        [ theta_k1, thetadt_k1, Ptheta ] = kalman_pn_theta...
            (theta_k0, thetadt_k0, nz_theta, Ptheta_old, ...
            mac_perpL_k1, rng_k1);

    elseif LAW == 2 %DG filters
        % Current time step corrected estimate of range and its
        % time derivatives.
        [ rng_k1, rngdt_k1, rngdtdt_k1, Prng ] = kalman_dg_range...
```

```
                (rng_k0, rngdt_k0, rngdtdt_k0, nz_rng, Prng_old, ...
                mac_paraL_k1);

            % Current time step corrected estimate of LOS angle and its
            % time derivatives.
            [ theta_k1, thetadt_k1, thetadtdt_k1, Ptheta ] = ...
                kalman_dg_theta(theta_k0, thetadt_k0, thetadtdt_k0, ...
                nz_theta, Ptheta_old, mac_perpL_k1, rng_k1);

            % Adjust corrected estimates to include deterministic
            % acceleration inputs.
            thetadtdtplot_k1 = thetadtdt_k1 +
(mac_perpL_k1*GRAV/rng_k1);
                        % Update current time step corrected estimate
                        % of LOS angle acceleration (thetadtdt_k1)to
                        % include the deterministic acceleration input
                        % (missile's drag and boost accelerations
                        % perpendicular to the LOS). This term cannot
                        % be added inside the filter because changes
                        % in LOS angle acceleration are modeled by the
                        % filter as white noise.

            rngdtdtplot_k1 = rngdtdt_k1 + (mac_paraL_k1*GRAV);
                        % Update current time step corrected estimate
                        % of range acceleration (rangedtdt_k1) to
                        % include the deterministic acceleration input
                        % (missile's drag and boost accelerations
                        % parallel to the LOS). This term cannot be
                        % added inside the filter because changes in
                        % range acceleration are modeled by the filter
                        % as white noise.
        end

        % Determine the square root of the maximum eigenvalue of
        % the current time step corrected estimate covariance
        sqePtheta = sqrt(max(eig(Ptheta)));
        sqePrng = sqrt(max(eig(Prng)));

        % Generate current time step corrected estimate of missile
        % lead angle (thetaLM_k1) for guidance law.
        thetaLM_k1 = thetaVM - theta_k1;
                        % Corrected estimate of angle between LOS and
                        % VM for accel perpendicular to VM
    end

    % Missile guidance
    if LAW == 1         % PN guidance.

        if FILT == 0    % Actual scenario parameters given to guidance
                        % law in propnav.m.
            [ mag ] = propnav(thetaLM, thetadt, rngdt);
                        % PN guidance acceleration (g).

        else            % Kalman filter estimates given to guidance
```

```matlab
                        % law in propnav.m.

        if kk < 4    % Delays guidance acceleration until after
                     % Kalman filters have been initialized.
            mag = 0;% Sets guidance acceleration to zero.

        else
            [ mag ] = propnav(thetaLM_k1, thetadt_k1, rngdt_k1);
                     % PN guidance acceleration (g).
        end
    end
    % Limit guidance acceleration to MAXG to prevent excessive
    % drag and missile damage.
    if abs(mag) > PN_MAXG
        mag = sign(mag)*PN_MAXG; % Reassigns guidance acceleration
                                 % to the maximum value while still
                                 % retaining the sign.
    end
end

if LAW == 2         % DG guidance.

    if FILT == 0    % Actual scenario parameters given to guidance
                    % law in diffgeo.m. The previous time step
                    % missile acceleration vector (Ma) is used
                    % since the output of this function affects the
                    % current time step Ma. Using the previous time
                    % step Ma introduces some error into the
                    % guidance but since the filter is "OFF"
                    % (FILT = 0) the noise switch must also be
                    % "OFF" (NZ = 0). The guidance doesn't track
                    % with noise and no filter. With no noise, Ma
                    % is very smooth and the difference between Ma
                    % in one time step is rather small.

        [ mag ] = diffgeo(theta, thetadt, thetadtdt, ...
            rng, rngdt, rngdtdt, VM, Ma, thetaLM);
                    % DG guidance acceleration (g).

    else            % Kalman filter estimates given to guidance law
                    % in diffgeo.m. The previous time step missile
                    % acceleration vector (Ma) is used since the
                    % output of this function affects the current
                    % time step Ma. Using the previous time step Ma
                    % introduces some error into the guidance. With
                    % noise, Ma can be very rough and the
                    % difference between Ma in one time step can be
                    % large. To combat the error introduced, we
                    % have given the guidance law Kalman filter
                    % estimates from the previous time step so all
                    % information is from the same time step. This
                    % generates a guidance acceleration that is
                    % much more accurate but applied one time step
                    % late. The result is a much smoother guidance
```

94

```matlab
                      % output.

             if kk <= BST_TM/DELTA % Estimates are rough until the
                                   % Kalman filters have a some time to
                                   % settle. Delaying guidance
                                   % acceleration until after boost
                                   % results in much smoother guidance
                                   % output.

               mag = 0; % Sets guidance acceleration to zero.

             else
                 [ mag ] = diffgeo(theta_k1out(kk-1),...
                     thetadt_k1out(kk-1), thetadtdt_k1out(kk-1), ...
                     rng_k1out(kk-1), rngdt_k1out(kk-1), ...
                     rngdtdt_k1out(kk-1), VM_old, Ma, thetaLM_k0);
                         % DG guidance acceleration (g).

                 % Average guidance output
                 mag = (mag + magout(kk-1) + magout(kk-2) ...
                     + magout(kk-3))/4;
                         % Averages guidance output with three previous
                         % outputs. The result is a much smoother
                         % guidance output.
             end
         end
         if abs(mag) > DG_MAXG
             mag = sign(mag)*DG_MAXG; % Reassigns guidance acceleration
                                      % to the maximum value while still
                                      % retaining the sign.
         end
     end


     % Generate guidance vector
     Magu = cross([0;0;1],VMu);
                         % Guidance unit vector (Magu) is perpendicular
                         % to the missile velocity unit vector (VMu).
                         % It is always in x-y plane since both target
                         % and missile remain in x-y plane. The sign of
                         % guidance acceleration controls the direction
                         % of Magu.

     Mag = mag*Magu;     % Generates a guidance vector of appropriate
                         % length and direction.

     % Missile boost
     if kk < (BST_TM/DELTA)  % Boost phase is active.
         Mab = mab*VMu;      % Missile acceleration (g) due to boost.
                             % Boost thrust is always applied in the
                             % direction of the missile velocity unit
                             % vector (VMu).
         boost = 1;          % Maintains boost phase.
     else                    % Boost phase terminated.
```

95

```matlab
    Mab = [0; 0; 0];      % Boost no longer contributes to missile
                          % acceleration.
    boost = 0;            % Terminates boost phase.
end


% Missile Drag
% Determine current parasitic drag coefficient.
Fdp = fdp_value(malt, mspd, boost); % Parasitic drag force (N).


% Determine current induced drag coefficient.
Fdi = fdi_value(Mag, mspd, malt); % Induced drag force (N).


% Determine acceleration due to total drag (g).
mad = (Fdp + Fdi)/(MASS*GRAV);
                    % Magnitude of acceleration due to drag (mad) is
                    % due to parasitic and induced drag.
Mad = mad*(-VMu); % Acceleration due to drag is always applied in
                    % the opposite direction of the missile velocity
                    % unit vector (VMu).


% Calculate deterministic inputs for Kalman filters from
% LOS perpective
if kk >= 4  % Estimated theta and range are not available for
            % deterministic inputs until after the Kalman filters
            % are initialized.
    PL_k1 = [cos(theta_k1)*rng_k1; sin(theta_k1)*rng_k1; 0];
            % Kalman filters only have access to an estimate of the
            % LOS position vector (PL).
    PLu_k1 = PL_k1/norm(PL_k1);     % Estimated LOS position
                                    % unit vector.


    % Combined deterministic missile accelerations (g)
    % due to boost and drag.
    Mac = Mab + Mad;

    % Estimated magnitude of combined deterministic missile
    % acceleration parallel to the LOS (mac_paraL_k1) (g).
    mac_paraL_k1 = Mac'*PLu_k1;

    % Estsimated vector of combined deterministic missile
    % acceleration parallel to the LOS (Mac_paraL_k1) (g).
    Mac_paraL_k1 = mac_paraL_k1*PLu_k1;

    % Estsimated vector of combined deterministic missile
    % acceleration perpendicular to the LOS (Mac_perpL_k1) (g).
    Mac_perpL_k1 = Mac - Mac_paraL_k1;

    % Estimated magnitude of combined deterministic missile
    % acceleration perpendicular to the LOS (mac_perpL_k1) (g).
    mac_perpL_k1 = norm(Mac_perpL_k1);

    % Sign of mac_perpL_k1 gives direction and must be preserved.
    sign_mac_perpL_k1 = ...
```

96

```matlab
            sign(cross((Mac_perpL_k1/norm(Mac_perpL_k1)), PLu));

    % A positive mac_paraL_k1 from the missile's perspective
    % actually causes the LOS range to decrease.
    mac_paraL_k1 = -mac_paraL_k1;

    % Deterministic input mac_perpL_k1 causes theta to increase.
    if sign_mac_perpL_k1(3) < 0

    % A positive mac_perpL_k1 from the missile's perspective
    % actually causes the LOS angle to decrease.
        mac_perpL_k1 = -mac_perpL_k1;
    end
end

% Total missile acceleration
Ma = Mag + Mad + Mab; % Total missile accel in (g) due to Guidance
                      % (Mag), Drag (Mad), and Boost (Mab).

% Update output matrices
% Target position (m).
Tpout(1:3,kk) = Hp*T;
% Missile position (m).
Mpout(1:3,kk) = Hp*M;
% Missile guidance acceleration (g).
magout(1,kk) = mag;
% Missile drag acceleration (g).
madout(1,kk) = mad;
% Missile speed (m/s).
mspdout(1,kk) = mspd;
% Time (sec) . kk-1 is uesd because no zero index available.
tout(1,kk) = (kk-1)*DELTA;
% Actual LOS angle (rad).
thetaout(1,kk) = theta;
% Noisy LOS angle (rad).
nz_thetaout(1,kk) = nz_theta;
% Actual LOS angle rate (rad/sec).
thetadtout(1,kk) = thetadt;
% Actual LOS angle acceleration (rad/sec^2).
thetadtdtout(1,kk) = thetadtdt;

% Actual range to target (m).
rngout(1,kk) = rng;
% Noisy range to target (m).
nz_rngout(1,kk) = nz_rng;
% Actual range rate to target (m/sec).
rngdtout(1,kk) = rngdt;
% Actual range acceleration to target (m/sec^2).
rngdtdtout(1,kk) = rngdtdt;

if kk >= 4  % Available to update only after Kalman filters
            % have been initialized.
    % Current time step corrected estimate of LOS angle (rad).
    theta_k1out(1,kk) = theta_k1;
```

```matlab
            % Current time step corrected estimate of LOS angle
            % rate (rad/sec).
            thetadt_k1out(1,kk) = thetadt_k1;
            % Output matrix for sqrt of maximum eigenvalue of current
            % time step theta corrected estimate covariance.
            sqePthetaout(1,kk) = sqePtheta;

            % Current time step corrected estimate of range to target (m).
            rng_k1out(1,kk) = rng_k1;
            % Current time step corrected estimate of range rate to
            % target (m/sec).
            rngdt_k1out(1,kk) = rngdt_k1;
            % Output matrix for sqrt of maximum eigenvalue of current
            % time step range corrected estimate covariance.
            sqePrngout(1,kk) = sqePrng;

        if LAW == 2 % Available only if DG guidance is used.
            % Current time step corrected estimate of LOS angle
            % acceleration (rad/sec^2).
            thetadtdt_k1out(1,kk) = thetadtdtplot_k1;
            % Current time step corrected estimate of range
            % acceleration to target (m/sec^2).
            rngdtdt_k1out(1,kk) = rngdtdtplot_k1;
        end
end


% Stop criteria
% Actual range rate (rngdt) is opening after boost has
% been terminated.
if (rngdt > 0) && (kk > BST_TM/DELTA)
            % Time delay allows for boost to overcome drag and
            % accelerate missile speed above target speed.
    break   % Stops simulation.
end

% 2)Missile speed is less than target speed
if (mspd < tspd) && (kk > BST_TM/DELTA)
            % Time delay allows for boost to overcome drag and
            % accelerate missile speed above target speed.
    break   % Stops simulation.
end

% 3) Range to target is less than 5m
if (rng < 5)
    break% Stops simulation.
end

% Update filter variables
thetadt_old = thetadt;  % Actual LOS angle rate (rad/sec).
rngdt_old = rngdt;      % Actual range rate (m/sec).
VM_old = VM;            % Actual missile velocity vector.
thetaLM_k0 = thetaLM_k1;% Estimate of angle (rad) between missile
                        % velocity vector and x-axis.
```

98

```matlab
    if kk >= 4  % Available to update only after Kalman filters
                % have been initialized.
        theta_k0 = theta_k1;      % Estimate of LOS angle (rad).
        thetadt_k0 = thetadt_k1;% Estimate of LOS angle rate (rad/sec).
        Ptheta_old = Ptheta;      % Theta corrected estimate covariance.

        rng_k0 = rng_k1;          % Estimate of range (m).
        rngdt_k0 = rngdt_k1;      % Estimate of range rate (m/sec).
        Prng_old = Prng;          % Range corrected estimate covariance.

        if LAW == 2 % Available only if DG guidance is used.
            thetadtdt_k0 = thetadtdt_k1;% Estimate of LOS angle
                                        % acceleration (rad/sec^2).
            rngdtdt_k0 = rngdtdt_k1;    % Estimate of range
                                        % acceleration (m/sec^2)
        end
    end

    % Missile Motion
    M = missile_motion(M, Ma);  % Returns updated missile state vector.

    % Target motion
    if time_to_impact(rng, rngdt) < TRN_TM  % Initiates turn when
                                            % impact is TURN_TIME
                                            % seconds away
        turn = 1;                 % Turn value is 0 when target motion
                                  % is straight, 1 when turning.
    end

    T = target_motion( T, turn );% Returns updated target state vector.
end % End primary loop

%Update output matrices by removing unused columns.
% Target position (m).
Tpout = Tpout(1:3,1:kk);
% Missile position (m).
Mpout = Mpout(1:3,1:kk);
% Missile guidance acceleration (g).
magout = magout(1,1:kk);
% Missile drag acceleration (g).
madout = madout(1,1:kk);
% Missile speed (m/sec^2).
mspdout = mspdout(1,1:kk);
% Time (sec).
tout = tout(1,1:kk);

% Actual LOS angle (rad).
thetaout = thetaout(1,1:kk);
% Noisy LOS angle (rad).
nz_thetaout = nz_thetaout(1,1:kk);
% Actual LOS angle rate (rad/sec).
thetadtout = thetadtout(1,1:kk);
% Actual LOS angle acceleration (rad/sec^2).
thetadtdtout = thetadtdtout(1,1:kk);
```

```matlab
% Current time step corrected estimate of LOS angle (rad).
theta_k1out = theta_k1out(1,1:kk);
% Current time step corrected estimate of LOS angle rate (rad/sec).
thetadt_k1out = thetadt_k1out(1,1:kk);
% Current time step corrected estimate of LOS angle
% acceleration (rad/sec^2).
thetadtdt_k1out = thetadtdt_k1out(1,1:kk);
% Output matrix for sqrt of maximum eigenvalue of current time step
% theta corrected estimate covariance.
sqePthetaout = sqePthetaout(1,1:kk);


% Actual range to target (m).
rngout = rngout(1,1:kk);
% Noisy range to target (m).
nz_rngout = nz_rngout(1,1:kk);
% Actual range rate to target (m/sec).
rngdtout = rngdtout(1,1:kk);
% Actual range acceleration to target (m/sec^2).
rngdtdtout = rngdtdtout(1,1:kk);
% Current time step corrected estimate of range to target (m).
rng_k1out = rng_k1out(1,1:kk);
% Current time step corrected estimate of range rate to target (m/sec).
rngdt_k1out = rngdt_k1out(1,1:kk);
% Current time step corrected estimate of range acceleration to
% target (m/sec^2).
rngdtdt_k1out = rngdtdt_k1out(1,1:kk);
% Output matrix for sqrt of maximum eigenvalue of current time step
% range corrected estimate covariance.
sqePrngout = sqePrngout(1,1:kk);


% Generate error arrays
% Difference between actual and noisy LOS angle (rad).
theta_err = thetaout - nz_thetaout;
% Difference between actual and estimated LOS angle (rad).
theta_k1err = thetaout - theta_k1out;
% Difference between actual and estimated LOS angle rate (rad/sec).
thetadt_k1err = thetadtout - thetadt_k1out;
% Difference between actual and estimated LOS angle
% acceleration (rad/sec^2).
thetadtdt_k1err = thetadtdtout - thetadtdt_k1out;


% Difference between actual and noisy range (m).
rng_err = rngout - nz_rngout;
% Difference between actual and estimated range (m).
rng_k1err = rngout - rng_k1out;
% Difference between actual and estimated range rate (m/sec).
rngdt_k1err = rngdtout - rngdt_k1out;
% Difference between actual and estimated range
% acceleration (rad/sec^2).
rngdtdt_k1err = rngdtdtout - rngdtdt_k1out;


%Plotting Results
    if PLOTS == 1 % Plotting switch turns off plots when PLOTS == 0.
    screen_size = get(0,'ScreenSize');
```

```matlab
% Overall Scenario Plots
figure(1)
subplot(2, 3, 1);
plot(tout, rngout)
title(['Range to Target (m), Miss Distance = ' ...
    num2str(min(rngout))]);
xlabel('Time (sec)');
ylabel('Range (m)');

subplot(2, 3, 2);
plot(tout, mspdout)
title('Missile Speed (m/s)');
xlabel('Time (sec)');
ylabel('Speed (m/s)');

subplot(2, 3, 3);
plot(tout, thetaout*180/pi)
title('Theta (LOS Angle wrt x-axis) (deg)');
xlabel('Time (sec)');
ylabel('Theta (deg)');

subplot(2, 3, 4);
plot(tout, magout)
title('Magnitude of Missile Guidance (g)');
xlabel('Time (sec)');
ylabel('Guidance (g)');
axis([0 ((kk*DELTA) + 2) -30 30]);

subplot(2, 3, 5);
plot(tout, madout)
title('Magnitude of Missile Drag (g)');
xlabel('Time (sec)');
ylabel('Accel (g)');

subplot(2, 3, 6);
plot(Tpout(1, :), Tpout(2, :), 'r-', 'LineWidth', 2)
hold on
plot(Mpout(1, :),Mpout(2, :), 'b-')
title('Encounter Geometry in the x-y plane');
axis('equal');
xlabel('x-axis');
ylabel('y-axis');

f1 = figure (1);
set(f1, 'Position', [0 0 screen_size(3) screen_size(4)]);
                        % Enlarges plot to full screen.

%Theta filter plots
figure(2)
subplot(2, 3, 1);
plot(tout, nz_thetaout*180/pi,'g')
hold on
plot(tout, theta_k1out*180/pi,'r')
```

```matlab
plot(tout, thetaout*180/pi)
title('Noisy, Est and Actual Theta (deg)');
xlabel('Time (sec)');
ylabel('Theta (deg)');
legend('Noisy Theta', 'Estimated Theta', 'Actual Theta');

subplot(2, 3, 4);
plot(tout, theta_err*180/pi, '.')
hold on
plot(tout, sqePthetaout*180/pi, 'g');
plot(tout, theta_k1err*180/pi, 'r')
title('Measured and Est Theta Error (deg)');
xlabel('Time (sec)');
ylabel('Error (deg)');
legend('Measured Theta Error', 'Estimated SqePtheta',...
    'Estimated Theta Error');
axis([0 ((kk*DELTA) + 2) 0 0.5])

subplot(2, 3, 2);
plot(tout, thetadt_k1out*180/pi, 'r');
hold on
plot(tout, thetadtout*180/pi);
title('Actual and Est Thetadot (deg/sec)');
xlabel('Time (sec)');
ylabel('Thetadot (deg/sec)');
legend('Estimated Thetadot','Actual Thetadot');
axis([0 ((kk * DELTA) + 2) -2 2])

subplot(2, 3, 5);
plot(tout, thetadt_k1err*180/pi, 'r');
title('Estimated Thetadt Error (deg/sec)');
xlabel('Time (sec)');
ylabel('Error (deg/sec)');
legend('Est Thetadt Error');
axis([0 ((kk*DELTA) + 2) -2 2])

subplot(2, 3, 3);
plot(tout, thetadtdtout*180/pi);
title('Actual Thetadtdt (deg/sec^2)');
xlabel('Time (sec)');
ylabel('Thetadtdt (deg/sec^2)');
legend('Actual Thetadtdt');
axis([0 ((kk*DELTA) + 2) -15 15]);

if LAW == 2
    subplot(2, 3, 3);
    plot(tout, thetadtdt_k1out*180/pi, 'r');
    hold on
    plot(tout, thetadtdtout*180/pi);
    title('Actual and Est Thetadtdt (deg/sec^2)');
    xlabel('Time (sec)');
    ylabel('Thetadtdt (deg/sec^2)');
    legend('Estimated Thetadtdt','Actual Thetadtdt');
    axis([0 ((kk*DELTA) + 2) -15 15])
```

```matlab
        subplot(2, 3, 6);
        plot(tout, thetadtdt_k1err*180/pi, 'r');
        title('Estimated Thetadtdt Error (deg/sec^2)');
        xlabel('Time (sec)');
        ylabel('Thetadtdt (deg/sec^2)');
        legend('Estimated Thetadtdt Error');
        axis([0 ((kk*DELTA) + 2) -20 20])
end
f2 = figure (2);
set(f2, 'Position', [0 0 screen_size(3) screen_size(4)]);
                        % Enlarges plot to full screen.


%Range filter plots
figure(3)
subplot(2, 3, 1);
plot(tout, nz_rngout, 'g')
hold on
plot(tout, rng_k1out,'r')
plot(tout, rngout)
title('Actual and Est Range (m)');
xlabel('Time (sec)');
ylabel('Range (m)');
legend('Noisy Range', 'Estimated Range', 'Actual Range');

subplot(2, 3, 4);
plot(tout, rng_err, '.')
hold on
plot(tout, sqePrngout, 'g');
plot(tout, rng_k1err, 'r');
title('Measured and Est Range Error (m)');
xlabel('Time (sec)');
ylabel('Error (m)');
legend('Measured Range Error', 'Estimated SqePrange',...
    'Estimated Range Error');
axis([0 ((kk*DELTA) + 2) 0 100])

subplot(2, 3, 2);
plot(tout, rngdt_k1out, 'r');
hold on
plot(tout, rngdtout);
title('Actual and Est Range Rate (m/sec)');
xlabel('Time (sec)');
ylabel('Range Rate (m/sec)');
legend('Estimated Range Rate','Actual Range Rate');
axis([0 ((kk*DELTA) + 2) -1200 100])

subplot(2, 3, 5);
plot(tout, rngdt_k1err);
title('Estimated Rngdt Error (m)');
xlabel('Time (sec)');
ylabel('Error (m/sec)');
legend('Est Rngdt Error');
axis([0 ((kk*DELTA) + 2) -100 100])
```

103

```matlab
    subplot(2, 3, 3);
    plot(tout, rngdtdtout);
    title('Actual Rngdtdt (deg/sec^2)');
    xlabel('Time (sec)');
    ylabel('Rngdtdt (m/sec^2)');
    legend('Actual Rngdtdt');
    axis([0 ((kk*DELTA) + 2) -200 200])

    if LAW == 2
        subplot(2, 3, 3);
        plot(tout, rngdtdt_k1out, 'r');
        hold on
        plot(tout, rngdtdtout);
        title('Actual and Est Rngdtdt (deg/sec^2)');
        xlabel('Time (sec)');
        ylabel('Rngdtdt (m/sec^2)');
        legend('Estimated Rngdtdt','Actual Rngdtdt');
        axis([0 ((kk*DELTA) + 2) -500 500])

        subplot(2, 3, 6);
        plot(tout, rngdtdt_k1err, 'r');
        title('Estimated Rngdtdt Error(deg/sec^2)');
        xlabel('Time (sec)');
        ylabel('Rngdtdt (m/sec^2)');
        legend('Estimated Rngdtdt Error');
        axis([0 ((kk*DELTA) + 2) -100 100])

    end

    f3 = figure (3);
    set(f3,'Position',[0 0 screen_size(3) screen_size(4)]);
                        % Enlarges plot to full screen.

end
```

```
% MAX_NOISE
% Generates a plot of the maximum noise factor vs. LOS angle.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%    File:               max_noise.m
%    Name:               LT Adam Osborn
%    Component Runtime:  8.1 (R2013a)
%    Compiler:           4.18.1 (R2013a)
%                        32-bit (Windows XP)
%    Date:               06 February 2014
%    Description:        Runs multiple simulations using sim_kb.m to
%                        determine the maximum amount of noise the
%                        applied guidance can withstand while still
%                        maintaining 70% effectiveness.
%    Inputs:             Vector of maximum effective missile range with
%                        no noise (rho) (m) generated by kb_generator.m.
%    Outputs:            Plot of maximum noise multiplier vs. LOS angle.
%    Process:
%    Assumptions:
%    Comments:           Simulations with increasing noise factors are
%                        run at a constant target range. This test range
%                        is 10% below maximum effective missile range
%                        with no noise at the corresponding aspect
%                        angle. Adjusting NZ_FACTOR global value also
%                        affects the variance of the simulated LOS angle
%                        and range sensors.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%------ Initialize MATLAB® Workspace ------
format long;     % Scaled fixed point display format for all float
                 % variables with 15 digits for double and 7 digits
                 % for single.

% close all;     % Closes all previous figures.
%
% clear all;     % Clears all previous workspace variables.

init;            % Runs init.m to load global workspace variables.

% Load noiseless range from simulations with RUNS = 10.
 load 'Final_Plotting_Workspace.mat';
 if LAW == 1
     rho = PN_no_noise;
 elseif LAW ==2
     rho = DG_no_noise;
 end

%------ define globals ------
global DEGSTEP INITSPD ALT RUNS NZ_FACTOR SIG_RNG SIG_THETA

%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
% Start timer for kinematic boundary generation time.
```

105

```matlab
tic;

% Initialize current maximum noise factor matrix
max_factor = zeros(((180/DEGSTEP)+1),1);

%------ functions ------
% Evaluate maximum effective noise factor at each aspect angle
for kk = 1:((180/DEGSTEP)+1)     % One cycle for each aspect angle.

    % Determine target state variables for this aspect angle
    T_hdg = (kk-1)*DEGSTEP;       % Set target heading to aspect angle
    x_spd = INITSPD*cos(T_hdg*pi/180);  % Component of target speed
                                        % along x-axis.
    y_spd = INITSPD*sin(T_hdg*pi/180);  % Component of target speed
                                        % along y-axis.
    T_rng = 0.9 * rho(kk);       % Reduce noiseless range by 10%

    % Set initial target state
    T_init = [T_rng;x_spd;0;y_spd;ALT;0];

    % First test loop (step size = 10)
    for NZ_FACTOR = 10 : 10 : 100
        disp(['*** Noise Factor =',...
            num2str(NZ_FACTOR),', step size = 10 ***'])

        %Reset variables
        misses = 0;
        hits = 0;
        swings = 0;

        % Update sensor variance
        SIG_THETA = NZ_FACTOR*0.001;    % LOS angle sensor uncertainty
                                        % (rad) given in Pehr thesis
                                        % pg. 16.
        SIG_RNG = NZ_FACTOR*10;         % Range sensor uncertainty (m)
                                        % given in Pehr thesis pg. 16.

        % Determine if missile is effective at this noise level
        while swings <= RUNS

            % Run simulation
            [rngout] = sim_kb(T_init);

            % Determine if missile hit
            if min(rngout)<=5
                disp(['>>> HIT, Heading ',...
                    num2str(T_hdg),' deg, Noise Factor = ',...
                    num2str(NZ_FACTOR), ' <<<'])

                hits = hits + 1;
            else
                disp(['>>> MISS, Heading ',...
                    num2str(T_hdg),' deg, Noise Factor = ',...
```

```matlab
                num2str(NZ_FACTOR), ' <<<'])

            misses = misses + 1;
        end

        % Missile is ineffective if it misses 30% of the time
        if misses == 0.3*RUNS;
            noise_factor = NZ_FACTOR-10; % Save missile's longest
                                         % effective range.
            break
        end

        % Missile is effective if it hits 70% of the time
        if hits == 0.7*RUNS;
            noise_factor = NZ_FACTOR;    % Save missile's longest
                                         % effective range.
            break
        end

        swings = swings + 1; % Increment count of simulations
    end

    % If missile is ineffective, move to the next test loop
    if misses == 0.3*RUNS;
        break
    end
end

% Second test loop (step size = 1)
for NZ_FACTOR = noise_factor+1 : 1 : noise_factor+9
    disp(['*** Noise Factor =',...
        num2str(NZ_FACTOR),', step size = 1 ***'])

    %Reset variables
    misses = 0;
    hits = 0;
    swings = 0;

    % Update sensor variance
    SIG_THETA = NZ_FACTOR*0.001;    % LOS angle sensor uncertainty
                                    % (rad) given in Pehr thesis
                                    % pg. 16.
    SIG_RNG = NZ_FACTOR*10;         % Range sensor uncertainty (m)
                                    % given in Pehr thesis pg. 16.

    % Determine if missile is effective at this noise level
    while swings <= RUNS

        % Run simulation
        [rngout] = sim_kb(T_init);

        % Determine if missile hit
        if min(rngout)<=5
```

```matlab
                disp(['>>> HIT, Heading ',...
                    num2str(T_hdg),' deg, Noise Factor = ',...
                    num2str(NZ_FACTOR), ' <<<'])

                hits = hits + 1;
            else
                disp(['>>> MISS, Heading ',...
                    num2str(T_hdg),' deg, Noise Factor = ',...
                    num2str(NZ_FACTOR), ' <<<'])

                misses = misses + 1;
            end

            % Missile is ineffective if it misses 30% of the time
            if misses == 0.3*RUNS;
                noise_factor = NZ_FACTOR-1; % Save missile's longest
                                            % effective range.
                break
            end

            % Missile is effective if it hits 70% of the time
            if hits == 0.7*RUNS;
                noise_factor = NZ_FACTOR;   % Save missile's longest
                                            % effective range.
                break
            end

            swings = swings + 1; % Increment count of simulations
        end

    % If missile is ineffective, move to the next test loop
    if misses == 0.3*RUNS;
        break
    end
end

    % Second test loop (step size = 0.1)
for NZ_FACTOR = noise_factor+0.1 : 0.1 : noise_factor + 0.9
    disp(['*** Noise Factor =',num2str(NZ_FACTOR),...
        ', step size = 0.1 ***'])

    %Reset variables
    misses = 0;
    hits = 0;
    swings = 0;

    % Update sensor variance
    SIG_THETA = NZ_FACTOR*0.001;     % LOS angle sensor uncertainty
                                     % (rad) given in Pehr thesis
                                     % pg. 16.
    SIG_RNG = NZ_FACTOR*10;          % Range sensor uncertainty (m)
                                     % given in Pehr thesis pg. 16.
```

```matlab
            % Determine if missile is effective at this noise level
            while swings <= RUNS

                % Run simulation
                [rngout] = sim_kb(T_init);

                % Determine if missile hit
                if min(rngout)<=5
                    disp(['>>> HIT, Heading ',...
                        num2str(T_hdg),' deg, Noise Factor = ',...
                        num2str(NZ_FACTOR), ' <<<'])

                    hits = hits + 1;
                else
                    disp(['>>> MISS, Heading ',...
                        num2str(T_hdg),' deg, Noise Factor = ',...
                        num2str(NZ_FACTOR), ' <<<'])

                    misses = misses + 1;
                end

                % Missile is ineffective if it misses 30% of the time
                if misses == 0.3*RUNS;
                    noise_factor = NZ_FACTOR-0.1; % Save missile's longest
                                                  % effective range.
                    break
                end

                % Missile is effective if it hits 70% of the time
                if hits == 0.7*RUNS;
                    noise_factor = NZ_FACTOR;     % Save missile's longest
                                                  % effective range.
                    break
                end

                swings = swings + 1; % Increment count of simulations
            end

            % If missile is ineffective, move to the next test loop
            if misses == 0.3*RUNS;
                break
            end
        end

    % Update max_factor plotting vector with maximum noise
    % factor at this aspect angle
    max_factor(kk) = noise_factor;

end

% Stop timer for noise factor plot generation time.
toc
```

109

```
% Generate theta vector for plotting
theta = pi/180*(0:DEGSTEP:180);

% Plot maximum noise factor vs. aspect angle
figure(1)
plot(theta, max_factor)
```

B.   SIMULATION GUIDANCE LAW FILES

```
function [ mag ] = propnav (thetaLM_k1, thetadt_k1, rng_dt)
% PROPNAV
% Computes the PN guidance acceleration vector.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%    File:              propnav.m
%    Name:              LT Adam Osborn
%    Component Runtime: 8.1 (R2013a)
%    Compiler:          4.18.1 (R2013a)
%                       32-bit (Windows XP)
%    Date:              06 February 2014
%    Description:       Uses the proportional navigation guidance law
%                       to compute the magnitude of the missile
%                       guidance acceleration vector (mag).
%    Inputs:            Missile lead angle estimate (thetaLM_k1) (rad)
%                       based on the Kalman filter's current time step
%                       estimate of LOS angle (theta_k1) (rad).
%                       Kalman filter's current time step estimates of:
%                       LOS angle rate (thetadt_k1) (rad/sec)
%                       range rate (rngdt_k1) (m/sec)
%    Outputs:           Magnitude of the missile guidance acceleration
%                       vector (mag) (g).
%    Process:
%    Assumptions:
%    Comments:
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%------ define globals ------
global NPRM GRAV
%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
%------ functions ------
% Magnitude of missile guidance acceleration vector (g)
mag = NPRM*(-rng_dt)*thetadt_k1/(cos(thetaLM_k1) * GRAV);

end
```

```matlab
function [ mag ] = diffgeo (theta_k1, thetadt_k1, thetadtdt_k1,...
    rng_k1, rngdt_k1, rngdtdt_k1, VM, Ma, thetaLM_k1)
% DIFFGEO
% Computes the DG guidance acceleration vector.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:               diffgeo.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Uses the differential geometry guidance law to
%                       compute the magnitude of the missile guidance
%                       acceleration vector (Mag).
%   Inputs:             Missile velocity vector (VM) (m/sec)
%                       Missile acceleration vector (Ma) (g)
%                       Missile lead angle estimate (thetaLM_k1) (rad)
%                       based on the Kalman filter's current time step
%                       estimate of LOS angle (theta_k1) (rad).
%                       Kalman filter's current time step estimates of:
%                       LOS angle (theta_k1) (rad)
%                       LOS angle rate (thetadt_k1) (rad/sec)
%                       LOS angle acceleration
%                           (thetadtdt_k1) (rad/sec^2)
%                       range (rng_k1) (m)
%                       range rate (rngdt_k1) (m/sec)
%                       range acceleration (rngdtdt_k1) (m/sec^2)
%   Outputs:            Magnitude of the missile guidance acceleration
%                       vector (mag) (g).
%   Process:            Differential geometry guidance requires the
%                       magnitude of the target's acceleration vector
%                       (Ta) as well as the target's lead angle
%                       (thetaLT). The target lead angle is defined as
%                       the angle from the target's velocity vector
%                       (VT) to the LOS position vector (PL).
%                       The Kalman filter provides an estimate of the
%                       velocity and acceleration of the LOS. Missile
%                       velocity and acceleration effects are
%                       removed leaving an estimate of the target's
%                       velocity and acceleration. These estimates are
%                       used to calculate the DG guidance acceleration
%                       for the missile.
%   Assumptions:        The actual missile velocity vector (VM) and
%                       acceleration vector (Ma) are assumed to be
%                       known.
%   Comments:           The missile acceleration vector (Ma) provided
%                       as an input is unavailable for the same
%                       discrete time step as the remaining inputs.
%                       This is because the DG guidance acceleration
%                       being calculated affects the missile
%                       acceleration vector (Ma). The previous time
%                       step missile acceleration vector is used. This
%                       introduces an error which is negligible when
%                       the output is smooth and the difference from
```

111

```
%                        one time step to the next is small. This is
%                        the case when the simulation is being run with
%                        no noise. For simulations with noise, the error
%                        introduced can be significant. To counter this
%                        error, all inputs are drawn from the previous
%                        time step for simulations with noise. This
%                        generates an output that is accurate, but
%                        applied one time step late. The effect on the
%                        missile guidance during simulations with noise
%                        has proved to be favorable to those with the
%                        significant error previously described.
%-------------------------------------------------------------------

%------ define globals ------
global NPRM GRAV
%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
%------ functions ------
%% Estimate the target's lead angle

% Estimate of the LOS position unit vector (PLu)
PL_k1 = [cos(theta_k1)*rng_k1; % Estimate of the LOS
         sin(theta_k1)*rng_k1; % position vector (PL).
                          0];
PLu_k1 = PL_k1/norm(PL_k1);     % Estimate of PLu.

%% Estimate of the target's velocity vector parallel to PLu.
% Estimate of the missile's velocity vector parallel to PLu.
VMpara_k1 = (VM'*PLu_k1)*PLu_k1;
% Estimate of the missile's velocity vector perpendicular to PLu.
VMperp_k1 = VM - VMpara_k1;
% Estimate of the target's velocity vector parallel to PLu.
VTpara_k1 = (rngdt_k1*PLu_k1) + VMpara_k1;

% Estimate of the LOS velocity vector perpendicular to PLu.
VLperp_k1 = thetadt_k1*rng_k1*(cross([0;0;1],PLu_k1));
% Estimate of the target's velocity vector perpendicular to PLu.
VTperp_k1 = VLperp_k1 + VMperp_k1;

% Estimate of the target's velocity vector
VT_k1 = VTpara_k1 + VTperp_k1;
% Estimate of the target's velocity unit vector
VTu_k1 = VT_k1/norm(VT_k1);

% Estimate of the target's lead angle
thetaLT_k1 = acos(PLu_k1'*VTu_k1);

%% Estimate the target's acceleration vector

% Convert the missile's acceleration vector to (m/sec^2)
AM = Ma*GRAV;

% Estimate of the missile's acceleration vector parallel to PLu.
```

```matlab
AMpara = (AM'*PLu_k1)*PLu_k1;
% Estimate of the target's acceleration vector parallel to PLu.
ATpara_k1 = (rngdtdt_k1*PLu_k1) + AMpara;

% Estimate of the missile's acceleration vector perpendicular to PLu.
AMperp = AM - AMpara;
% Estimate of the LOS acceleration vector perpendicular to PLu.
ALperp_k1 = thetadtdt_k1*rng_k1*(cross([0;0;1],PLu_k1));
% Estimate of the target's acceleration vector perpendicular to PLu.
ATperp_k1 = ALperp_k1 + AMperp;

% Estimate of the target's acceleration vector
AT_k1 = ATpara_k1 + ATperp_k1;

%% Magnitude of missile guidance acceleration vector (g)
mag = (norm(AT_k1)*cos(thetaLT_k1))/(cos(thetaLM_k1)*GRAV) + ...
    (NPRM*(-rngdt_k1)*thetadt_k1)/(cos(thetaLM_k1)*GRAV);
```

C.   SIMULATION FUNCTION FILES

```matlab
function [ tgo ] = time_to_impact ( rng, rngdt )
% TIME_TO_IMPACT
% Computes time to impact with target.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%    File:              time_to_impact.m
%    Name:              LT Adam Osborn
%    Component Runtime: 8.1 (R2013a)
%    Compiler:          4.18.1 (R2013a)
%                       32-bit (Windows XP)
%    Date:              06 February 2014
%    Description:       Computes time remaining until impact (tgo) with
%                       target based on range and range rate.
%    Inputs:            LOS range (rng) (m)
%                       LOS range rate (rngdt) (m/sec)
%    Outputs:           Time remaining until impact (tgo) (sec).
%    Process:
%    Assumptions:
%    Comments:          Portions of this code have been reused from
%                       Pehr's Thesis.


%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%------ define globals ------
%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
%------ functions ------

if (rngdt == 0) % Prevent dividing by zero when rngdt = 0.
    tgo = 100;  % Choose some large number to prevent turn at the
                % beginning of the simulation. Simulation ends if
```

113

```matlab
                        % rngdt goes positive.
else
    tgo = rng / (-rngdt);    % Range rate should always be less than or
                              % equal to zero.
end
end
```

```matlab
function [ mach_speed ] = mach_speed ( malt )
% MACH_SPEED
% Computes Mach speed (m/sec) for a given altitude.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%    File:              mach_speed.m
%    Name:              LT Adam Osborn
%    Component Runtime: 8.1 (R2013a)
%    Compiler:          4.18.1 (R2013a)
%                       32-bit (Windows XP)
%    Date:              06 February 2014
%    Description:       Computes the linear approximation of Mach 1
%                       velocity (m/sec) for a given altitude based on
%                       standard ICAO atmosphere.
%    Inputs:            Missile altitude (malt) (m)
%    Outputs:           Velocity of Mach 1 (mach_speed) (m/sec)
%    Process:           Uses polynomial fit of the altitude/velocity
%                       curve of Mach 1 in a standard ICAO atmosphere.
%    Assumptions:
%    Comments:          Portions of this code have been reused from
%                       Pehr's Thesis.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%------ define globals ------
%------ define constants ------
% Constants for altitude/velocity curve in standard ICAO atmosphere.
A = [-0.0041 340.3];    % Altitudes below 11km.
B = 295.1;              % Altitude of 11-20km.
C = [0.00067 281.7];    % Altitudes greater than 20km.

%------ define input vector ------
%------ initialize variables ------
malt = abs(malt);    % Absolute value accounts for NED coords

%------ functions ------
if (malt<11000) % Use A variables if altitude is below 11km.
    mach_speed = polyval(A,malt); % Velocity of Mach 1 (m/sec).

elseif (malt>20000) % Use B variables if altitude is above 20km.
    mach_speed = polyval(C,malt); % Velocity of Mach 1 (m/sec).

else
    mach_speed = B;    % Velocity of Mach 1 (m/sec).
end

end
```

```matlab
function [ rho ] = rho_value ( malt )
% RHO_VALUE
% Computes the atmospheric density.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:               rho_value.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Computes the atmospheric density at the given
%                       altitude for ICAO standard atmosphere.
%   Inputs:             Missile altitude (malt) (m)
%   Outputs:            Atmospheric density (rho) (kg/m^3)
%   Process:
%   Assumptions:
%   Comments:           Portions of this code have been reused from
%                       Pehr's Thesis.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%------ define globals ------
%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
malt = abs(malt);   % Absolute value accounts for NED coordinates.

%------ functions ------
if malt > 9144  % Atmospheric density below 9144 meters.
    rho = 1.75228763*exp(-malt/6705.6); % Atmospheric density
                                        % (kg/m^3).

else    % Atmospheric density above or at 9144 meters.
    rho = 1.22557*exp(-malt/9144);      % Atmospheric density
                                        % (kg/m^3).
end

end
```

```matlab
function [ Cdp ] = cdp_value (mach, boost)
% CDP_VALUE
% Computes approximation of zero lift drag coefficient.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%   File:                cdp_value.m
%   Name:                LT Adam Osborn
%   Component Runtime:   8.1 (R2013a)
%   Compiler:            4.18.1 (R2013a)
%                        32-bit (Windows XP)
%   Date:                06 February 2014
%   Description:         Computes approximation of zero lift drag
%                        coefficient from graph of Cdp vs. Mach number.
%   Inputs:              Missile mach number (mach) (unitless)
%                        Boost value (boost) (1 during "boost phase", 0
%                        otherwise)
%   Outputs:             Parasitic drag coefficient (Cdp) (unitless)
%   Process:             Uses polynomial fit of the parasitic drag
%                        coefficient curve given in Pehr thesis pg. 18.
%   Assumptions:
%   Comments:            Broadston used a similar curve but experienced
%                        large transients transitioning out of
%                        supersonic speed (below Mach 1). Pehr modified
%                        the parasitic drag coefficient curve by
%                        reducing the slope of the curve at Mach 1 which
%                        smoothed the transition. This modification is
%                        described in Pehr's thesis pg. 18.
%                        Portions of this code have been reused from
%                        Pehr's Thesis.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%------ define globals ------
%------ define constants ------
% Constants for the parasitic drag coefficient curve
NoBoost = [-0.0014 0.0299 -0.2110 0.6256];   % Curve when boost
                                             % value is 0.
Boost = [-0.0012 0.0243 -0.1521 0.4044];     % Curve when boost
                                             % value is 1.

%------ define input vector ------
%------ initialize variables ------
%------ functions ------
if (boost & (mach<0.7))
    Cdp=0.15;
end

if (~boost & (mach<0.7))
    Cdp=0.25;
end

if (boost & (mach>=0.7) & (mach<1.2))
    Cdp=(mach-0.7)*0.2 + 0.15;
end

if (~boost & (mach>=0.7) & (mach<1.2))
```

117

```
    Cdp=(mach-0.7)*0.3 + 0.25;
end

if ((mach>=1.2) & (boost~=0))
    Cdp=polyval(Boost, mach);
end

if ((mach>=1.2) & (boost==0))
    Cdp=polyval(NoBoost, mach);
end

if ((mach>5 & boost))
    Cdp=0.10;
end

if ((mach>6.4) & ~boost)
    Cdp=0.132;
end

end
```

```matlab
function [ Fdp ] = fdp_value (malt, mspd, boost)
% FDP_VALUE
% Computes missile's parasitic drag force.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%   File:                 fdp_value.m
%   Name:                 LT Adam Osborn
%   Component Runtime:    8.1 (R2013a)
%   Compiler:             4.18.1 (R2013a)
%                         32-bit (Windows XP)
%   Date:                 06 February 2014
%   Description:          Computes the parasitic drag force for a missile
%                         with frontal area SREF in standard atmosphere.
%   Inputs:               Missile altitude (malt) (m)
%                         Missile speed (mspd) (m/sec)
%                         Boost value (boost) (1 during "boost phase", 0
%                         otherwise)
%   Outputs:              Parasitic drag force (Fdp) (N)
%   Process:
%   Assumptions:
%   Comments:             Portions of this code have been reused from
%                         Pehr's Thesis.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%------ define globals ------
global SREF

%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
rho = rho_value(malt);          % Atmospheric density (kg/m^3).
mach = mspd/mach_speed(malt);   % Missile speed converted to mach value
                                % (unitless).
Q = rho*mspd^2/2;               % Dynamic pressure (kg*m/sec^2 or N)
Cdp = cdp_value(mach,boost);    % Parasitic drag coefficient
                                % (unitless).
%------ functions ------
%% Determine parasitic drag force
Fdp = Q*Cdp*SREF;               % Force due to parasitic drag (kg*m/s^2
or N)

end
```

```matlab
function [ Fdi] = fdi_value( Mag, mspd, malt )
% FDI_VALUE
% Computes missile's induced drag force.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%    File:              fdi_value.m
%    Name:              LT Adam Osborn
%    Component Runtime: 8.1 (R2013a)
%    Compiler:          4.18.1 (R2013a)
%                       32-bit (Windows XP)
%    Date:              06 February 2014
%    Description:       Computes the drag force on the missile induced
%                       by application of guidance acceleration.
%    Inputs:            Guidance acceleration vector (Mag) (g)
%                       Missile speed (mspd) (m/sec)
%                       Missile altitude (malt) (m)
%    Outputs:           Induced drag force (Fdp) (N)
%    Process:           Uses guidance acceleration forces (Mag) to
%                       determine the induced drag coefficients (Cdi)
%                       and subsequently use these coefficents to
%                       calculate the total induced drag force on
%                       the missile.
%    Assumptions:
%    Comments:          Induced drag is normally determined by the
%                       angle of attack. In this point mass model, we
%                       use normal forces to account for induced drag.
%                       The maximum subsonic parasitic drag coefficient
%                       (when boost is not applied) is used to
%                       approximate the subsonic induced drag
%                       coefficient as shown in Broadston's thesis
%                       pg. 19. Gravity is accounted for in this point
%                       mass model as an elevation guidance force.
%                       Portions of this code have been reused from
%                       Pehr's Thesis.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%------ define globals ------

%------ define input vector ------
%------ initialize variables ------
%------ functions ------
%------ define globals ------
global MASS SREF eAR GRAV

%------ define constants ------
Max_Cdp = 0.25; % The maximum subsonic parasitic drag coefficient when
                % boost is not applied. Used to determine the subsonic
                % induced drag coefficient

%------ initialize variables ------
rho = rho_value(malt);          % Atmospheric density (kg/m^3).
mach = mspd/mach_speed(malt);   % Missile speed converted to mach
                                % value (unitless).
Q = rho*mspd^2/2;               % Dynamic pressure (kg*m/sec^2 or N)
```

```matlab
%------ functions ------
%% Determine azimuthal and elevation guidance forces
% Missile azimuthal acceleration (m/sec^2).
Maa = sqrt(Mag(1)^2+ Mag(2)^2)*GRAV;
% Missile elevation acceleration (m/sec^2).
Mae = Mag(3)*GRAV;
% Azimuthal guidance force (kg*m/sec^2 or N).
Fa = MASS*Maa;
% Elevation guidance force (kg*m/sec^2 or N).
Fe = MASS*(Mae-GRAV);

%% Determine azimuthal and elevation normal force coefficients
Cna = Fa/(Q*SREF);  % Azimuthal normal coefficient (unitless).
Cne = Fe/(Q*SREF);  % Elevation normal coefficient (unitless).

%% Determine the overall induced drag coefficient
if (mach<1)      % Missile is subsonic.
    Cdi = Max_Cdp*sqrt(Fa^2+Fe^2)/(MASS*GRAV);

else            % Missile is supersonic.
    Cdi = (Cna^2+Cne^2)/(pi*eAR);   % Unitless induced drag coefficient

end

%% Determine induced drag force
Fdi = Cdi*Q*SREF;   % Force due to induced drag (kg*m/sec^2 or N).

end
```

```matlab
function [ nz_rng ] = noisy_range (rng)
% NOISY_RANGE
% Adds noise to range measurements.
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%   File:               noisy_range.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Adds white noise to range based on noise
%                       multiplier of randn.
%   Inputs:             Actual LOS range (rng) (m)
%   Outputs:            Simulated range sensor noisy LOS range
%                       measurement (nz_rng) (m)
%   Process:            Global variable NZ_FACTOR is used as a
%                       multiplier to the sensor accuracy (SIG_RNG).
%                       The range measurement (nz_rng) is generated by
%                       adding white noise with the randn function.
%   Assumptions:
%   Comments:
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%------ define globals ------
global SIG_RNG

%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
%------ functions ------
% Generate simulated range sensor noisy measurement (nz_rng).
nz_rng = rng + randn * SIG_RNG; % Add random zero mean
                                % gaussian white noise to
                                % actual range.

end
```

```matlab
function [ nz_theta ] = noisy_theta (theta)
% NOISY_THETA
% Adds noise to theta measurement.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:               noisy_theta.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Adds white noise to theta based on noise
%                       multiplier of randn.
%   Inputs:             Actual LOS angle (theta) (rad)
%   Outputs:            Simulated LOS angle sensor noisy LOS angle
%                       measurement (nz_theta) (rad)
%   Process:            Global variable NZ_FACTOR is used as a
%                       multiplier to the sensor accuracy (SIG_THETA).
%                       The LOS angle measurement (nz_theta) is
%                       generated by adding white noise with the
%                       randn function.
%   Assumptions:
%   Comments:
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%------ define globals ------
global SIG_THETA
%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
%------ functions ------
% Generate simulated LOS angle sensor noisy measurement (nz_theta).
nz_theta = theta + randn * SIG_THETA;
                        % Add random zero mean gaussian white noise
                        % to actual range.

end
```

123

```matlab
function [ M ] = missile_motion( M, Ma)
% MISSILE_MOTION
% Updates missile state vector (M).
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:               missile_motion.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Computes the updated missile state vector (M)
%                       including straight line motion and changes due
%                       to total acceleration (drag, guidance, boost).
%   Inputs:             Missile state vector (M)
%                       Missile total acceleration (Ma)
%   Outputs:            Updated Missile state vector (M)
%   Process:            The straight line motion transition matrix is
%                       used when turn value is 0. The turn transition
%                       matrix is used when turn value is 1.
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%------ define globals ------
global GRAV DELTA
%------ define constants ------
% Missile transition matrix for straight line motion.
Fm = [1, DELTA, 0,     0, 0,     0;
      0,     1, 0,     0, 0,     0;
      0,     0, 1, DELTA, 0,     0;
      0,     0, 0,     1, 0,     0;
      0,     0, 0,     0, 1, DELTA;
      0,     0, 0,     0, 0,     1];

%------ define input vector ------
%------ initialize variables ------
%------ functions ------
%Motion
xmzz = Ma*GRAV*(DELTA^2)/2; % Change in missile position (m) with
                           % constant acceleration.

vmzz = Ma*GRAV*DELTA;    %Change in missile velocity (m/sec) with
                           % constant acceleration.

% Total change in missile state other than straight line motion
Xmz = [xmzz(1), vmzz(1), xmzz(2), vmzz(2), xmzz(3), vmzz(3)]';

%Updated missile state vector
M = Fm*M + Xmz; % Adds straight line motion effects with the
                % accererlations due to drag, guidance, and boost.
end
```

```matlab
function [ T ] = target_motion( T, turn )
% TARGET_MOTION
% Updates target state vector (T).
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:              target_motion.m
%   Name:              LT Adam Osborn
%   Component Runtime: 8.1 (R2013a)
%   Compiler:          4.18.1 (R2013a)
%                      32-bit (Windows XP)
%   Date:              06 February 2014
%   Description:       Computes the updated target state vector (T)
%                      during straight line motion and changes due to
%                      turn acceleration. Turn is based on turn value.
%                      Target conducts turn toward missile of
%                      magnitude TRN_G (g) when turn value is 1.
%                      This occurs at TRN_TM (sec) from impact.
%                      TRN_TM is defined in init.m.
%   Inputs:            Target state vector (T)
%                      turn value (0 for no turn, 1 for turn)
%   Outputs:           Updated target state vector (T)
%   Process:           The straight line motion transition matrix is
%                      used when turn value is 0. The turn transition
%                      matrix is used when turn value is 1.
%   Assumptions:
%   Comments:
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%------ define globals ------
global TRN_G GRAV DELTA

%------ define constants ------
% Acceleration
a_turn = TRN_G*GRAV;     % Magnitude of TRN_G (g)
                         % converted by GRAV to (m/s^2).

%------ define input vector ------
%------ initialize variables ------
%------ functions ------
% Angular acceleration
w = a_turn/sqrt((T(2))^2 + (T(4))^2);   % Determined by dividing
                                        % tangential acceleration
                                        % (accel) by euclidean distance
                                        % of target from origin.
% Motion
if turn == 0    % Straight line motion from the beginning of the
                % simulation until turn value shifts to 1 when time to
                % impact (tgo) is less than or equal to TRN_TM from
                % init.m.

    % Target transition matrix for straight line motion.
    Ft = [1, DELTA, 0,      0, 0,      0;
          0,     1, 0,      0, 0,      0;
          0,     0, 1, DELTA, 0,      0;
          0,     0, 0,      1, 0,      0;
```

125

```matlab
              0,      0, 0,      0, 1, DELTA;
              0,      0, 0,      0, 0,      1];

    else    % Curved motion from tgo = TRN_TM (sec) to impact
            % with TRN_G (g) curvature.
        % Target transition matrix during a turn. Found in Prof. Hutchins
        % EC3320 notes Section 2 page 33. Modified for 3D translation.
        Ft = [1,       sin(w*DELTA)/w, 0, (1-cos(w*DELTA))/w, 0,      0;
              0,       cos(w * DELTA), 0,      -sin(w*DELTA), 0,      0;
              0, (1-cos(w*DELTA))/w, 1,      sin(w*DELTA)/w, 0,      0;
              0,        sin(w*DELTA), 0,        cos(w*DELTA), 0,      0;
              0,                  0, 0,                  0, 1, DELTA;
              0,                  0, 0,                  0, 0,      1];
    end

    T = Ft*T;    % Updated target state matrix after target motion
end
```

D.   SIMULATION FILTER FILES

```matlab
function [rng_k1, rngdt_k1, rngdtdt_k1, Prng_k1] = kalman_dg_range...
    (rng_k0, rngdt_k0, rngdtdt_k0, nz_rng, Prng_k0, mac_paraL_k1)
% KALMAN_DG_RANGE
% Generates an estimate and covariance of the three dimensional range
% state.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:               kalman_dg_range.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Kalman algorithm which generates current
%                       discrete time step corrected estimates of
%                       range, range rate, and range acceleration as
%                       well as the corrected covariance of the
%                       estimation.
%   Inputs:             Kalman filter's previous time step estimate of:
%                       range (rng_k0) (m)
%                       range rate (rngdt_k0) (m/sec)
%                       range acceleration (rngdtdt_k0) (m/sec^2)
%                       Kalman filter's previous time step corrected
%                       range covariance (Prng_k0)
%                       Noisy range measurement (nz_rng) (m)
%                       Estimated magnitude of combined deterministic
%                       missile acceleration parallel to the LOS
%                       (mac_paraL_k1).
%   Outputs:            Kalman filter's current time step estimate of:
%                       range (rng_k1) (m)
%                       range rate (rngdt_k1) (m/sec)
%                       range acceleration (rngdtdt_k1) (m/sec^2)
%                       Kalman filter's current time step corrected
%                       range covariance (Prng_k1)
%   Process:
%   Assumptions:
%   Comments:           Position and velocity portions of the missile's
%                       deterministic acceleration inputs are applied
%                       inside the filter. The acceleration portion is
%                       applied outside the filter since unpredictable
%                       accelerations are modeled as white noise inside
%                       the filter.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%------ define globals ------
global DELTA SIG_RNG Q2_DG_RNG GRAV TRN_G

%------ define constants ------
% Transition matrix
F = [1,DELTA,(DELTA^2)/2;
     0,    1,       DELTA;
     0,    0,           1];
```

127

```matlab
% Covariance matrix for uncorrelated range measurements
R = (SIG_RNG^2);

% Measurement matrix
H =[1, 0, 0];

% Plant noise gain matrix
G = eye(3);

%------ define input vector ------
% Set corrected range state values from previous discrete time step
xc = [     rng_k0;
          rngdt_k0;
       rngdtdt_k0];

Pc = Prng_k0;

%------ initialize variables ------
% Plant noise covariance multiplier
q2 = Q2_DG_RNG*3*DELTA*((TRN_G*GRAV)^2)/4; % Based on velocity variance

% Plant noise covariance
Q = q2*[(DELTA^5)/20, (DELTA^4)/8, (DELTA^3)/6;
         (DELTA^4)/8,  (DELTA^3)/3, (DELTA^2)/2;
         (DELTA^3)/6,  (DELTA^2)/2,       DELTA];

%------ functions ------
%% Prediction phase
% Predicted range state
u = (mac_paraL_k1)*GRAV*[(DELTA^2)/2; DELTA; 0];
                     % Position and velocity portions of the missile's
                     % deterministic acceleration inputs only.
xp = F*xc + G*u;     % Predicted range state

% Predicted range state estimation covariance
Pp = F*Pc*F' + Q;

%% Correction Phase
% Calculate Kalman Gain (W)
S = R + H*Pp*H';     % Innovation covariance
W = S\(Pp*H');       % A\b used vice inv(A)*b for speed of computation.

% Calculate the innovations
v = nz_rng - H*xp;

% Calculate corrected state vector (xc)
xc = xp + W*v;

% Calculate corrected covariance (Pc)
Pc = (eye(3)-W*H)*Pp*(eye(3)-W*H)' + W*R*W';

% Corrected range state values for current discrete time step
rng_k1 = xc(1);
```

```
rngdt_k1 = xc(2);
rngdtdt_k1 = xc(3);

Prng_k1 = Pc;

end
```

```matlab
function [theta_k1, thetadt_k1, thetadtdt_k1, Ptheta_k1] = ...
    kalman_dg_theta(theta_k0, thetadt_k0, thetadtdt_k0, nz_theta, ...
    Ptheta_k0, mac_perpL_k1, rng_k1)
% KALMAN_DG_THETA
% Generates an estimate and covariance of the three dimensional theta
% state.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:               kalman_dg_theta.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Kalman algorithm which generates current
%                       discrete time step corrected estimates of
%                       theta, theta rate, and theta acceleration as
%                       well as the corrected covariance of the
%                       estimation.
%   Inputs:             Kalman filter's previous time step estimate of:
%                       LOS angle (theta_k0) (rad)
%                       LOS angle rate (thetadt_k0) (rad/sec)
%                       LOS angle acceleration
%                           (thetadtdt_k0) (rad/sec^2)
%                       Kalman filter's previous time step corrected
%                       theta covariance (Ptheta_k0)
%                       Kalman filter's current time step estimate of
%                       range (rng_k1) (m)
%                       Noisy theta measurement (nz_theta) (rad)
%                       Estimated magnitude of combined deterministic
%                       missile acceleration perpendicular to the LOS
%                       (mac_perpL_k1).
%   Outputs:            Kalman filter's current time step estimate of:
%                       LOS angle (theta_k1) (rad)
%                       LOS angle rate (thetadt_k1) (rad/sec)
%                       LOS angle acceleration
%                           (thetadtdt_k1) (rad/sec^2)
%                       Kalman filter's current time step corrected
%                       theta covariance (Ptheta_k1)
%   Process:
%   Assumptions:
%   Comments:           Position and velocity portions of the missile's
%                       deterministic acceleration inputs are applied
%                       inside the filter. The acceleration portion is
%                       applied outside the filter since unpredictable
%                       accelerations are modeled as white noise inside
%                       the filter.
%-----------------------------------------------------------------------
-
%-----------------------------------------------------------------------
-
%------ define globals ------
global DELTA SIG_THETA Q2_DG_THETA1 Q2_DG_THETA2 GRAV TRN_G Q2_SHIFT
%------ define constants ------
% Transition matrix
```

130

```matlab
F = [1,DELTA,(DELTA^2)/2;
     0,    1,      DELTA;
     0,    0,         1];

% Covariance matrix for uncorrelated bearing measurements
R = (SIG_THETA^2);

% Measurement matrix
H =[1, 0, 0];

% Plant noise gain matrix
G = eye(3);

%------ define input vector ------
% Set corrected theta state values from previous discrete time step
xc = [    theta_k0;
        thetadt_k0;
      thetadtdt_k0];


Pc = Ptheta_k0;

%------ initialize variables ------
% Plant noise covariance multiplier
if rng_k1 >= Q2_SHIFT
    q2 = Q2_DG_THETA1*3*DELTA*...
        ((TRN_G*GRAV)^2)/(4*(rng_k1^2)); % Based on velocity variance
else
    q2 = Q2_DG_THETA2*3*DELTA*...
        ((TRN_G*GRAV)^2)/(4*(rng_k1^2)); % Based on velocity variance
end

% Plant noise covariance
Q = q2*[(DELTA^5)/20, (DELTA^4)/8, (DELTA^3)/6;
        (DELTA^4)/8,  (DELTA^3)/3, (DELTA^2)/2;
        (DELTA^3)/6,  (DELTA^2)/2,  DELTA];

%------ functions ------
%% Prediction phase
% Predicted theta state
u = [    atan2(mac_perpL_k1*GRAV*(DELTA^2), 2*rng_k1);
             4*mac_perpL_k1*GRAV*rng_k1*DELTA / ...
    (4*(rng_k1^2) + (mac_perpL_k1*GRAV)^2*(DELTA^4));
                                              0];
                % Position and velocity portions of the missile's
                % deterministic acceleration inputs only.
xp = F*xc + u;      % Predicted theta state

% Predicted range state estimation covariance
Pp = F*Pc*F' + Q;

%% Correction Phase
% Calculate Kalman Gain (W)
S = R + H*Pp*H';    % Innovation covariance
```

```matlab
W = S\(Pp*H'); % A\b used vice inv(A)*b for speed of computation.

% Calculate the innovations
v = nz_theta - H*xp;

% Calculate corrected state vector (xc)
xc = xp + W*v;

% Calculate corrected covariance (Pc)
Pc = (eye(3)-W*H)*Pp*(eye(3)-W*H)' + W*R*W';

% Corrected theta state values for current discrete time step
theta_k1 = xc(1);
thetadt_k1 = xc(2);
thetadtdt_k1 = xc(3);


Ptheta_k1 = Pc;


end
```

```matlab
function [rng_k1, rngdt_k1, Prng_k1] = kalman_pn_range...
    (rng_k0, rngdt_k0, nz_rng, Prng_k0, mac_paraL_k1)
% KALMAN_PN_RANGE
% Generates an estimate and covariance of the two dimensional range
% state.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:               kalman_pn_range.m
%   Name:               LT Adam Osborn
%   Component Runtime:  8.1 (R2013a)
%   Compiler:           4.18.1 (R2013a)
%                       32-bit (Windows XP)
%   Date:               06 February 2014
%   Description:        Kalman algorithm which generates current
%                       discrete time step corrected estimates of range
%                       and range rate as well as the corrected
%                       covariance of the estimation.
%   Inputs:             Kalman filter's previous time step estimate of:
%                       range (rng_k0) (m)
%                       range rate (rngdt_k0) (m/sec)
%                       Kalman filter's previous time step corrected
%                       range covariance (Prng_k0)
%                       Noisy range measurement (nz_rng) (m)
%                       Estimated magnitude of combined deterministic
%                       missile acceleration parallel to the LOS
%                       (mac_paraL_k1).
%   Outputs:            Kalman filter's current time step estimate of:
%                       range (rng_k1) (m)
%                       range rate (rngdt_k1) (m/sec)
%                       Kalman filter's current time step corrected
%                       range covariance (Prng_k1)
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
-
%-------------------------------------------------------------------------
-
%------ define globals ------

global DELTA SIG_RNG Q2_PN_RNG GRAV TRN_G
%------ define constants ------
% Transition matrix
F = [1, DELTA;
     0,     1];

% Covariance matrix for uncorrelated bearing measurements
R = (SIG_RNG^2);

% Measurement matrix
H =[1, 0];

% Plant noise gain matrix
G = eye(2);
```

133

```matlab
%------ define input vector ------
% Set corrected theta state values from previous discrete time step
xc = [    rng_k0;
        rngdt_k0];

Pc = Prng_k0;

%------ initialize variables ------
% Plant noise covariance multiplier
q2 = Q2_PN_RNG*3*((TRN_G*GRAV)^2)*DELTA/4;

% Plant noise covariance
Q = q2*[(DELTA^3)/3,  (DELTA^2)/2;
        (DELTA^2)/2,       DELTA];

%------ functions ------
%% Prediction phase
% Predicted range state
u = (mac_paraL_k1)*GRAV*[(DELTA^2)/2; DELTA];
                        % Position and velocity portions of the missile's
                        % deterministic acceleration inputs only.
xp = F*xc + G*u;      % Predicted range state

% Predicted range state estimation covariance
Pp = F*Pc*F' + Q;

%% Correction Phase
% Calculate Kalman Gain (W)
S = R + H*Pp*H';      % Innovation covariance
W = S\(Pp*H');        % A\b used vice inv(A)*b for speed of computation.

% Calculate the innovations
v = nz_rng - H*xp;

% Calculate corrected state vector (xc)
xc = xp + W*v;

% Calculate corrected covariance (Pc)
Pc = (eye(2)-W*H)*Pp*(eye(2)-W*H)' + W*R*W';

% Corrected range state values for current discrete time step
rng_k1 = xc(1);
rngdt_k1 = xc(2);

Prng_k1 = Pc;

end
```

```matlab
function [theta_k1, thetadt_k1, Ptheta_k1] = kalman_pn_theta...
    (theta_k0, thetadt_k0, nz_theta, Ptheta_k0, mac_perpL_k1, rng_k1)
% KALMAN_PN_THETA
% Generates an estimate and covariance of the two dimensional theta
% state.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%    File:              kalman_pn_theta.m
%    Name:              LT Adam Osborn
%    Component Runtime: 8.1 (R2013a)
%    Compiler:          4.18.1 (R2013a)
%                       32-bit (Windows XP)
%    Date:              06 February 2014
%    Description:       Kalman algorithm which generates current
%                       discrete time step corrected estimates of theta
%                       and theta rate as well as the corrected
%                       covariance of the estimation.
%    Inputs:            Kalman filter's previous time step estimate of:
%                       LOS angle (theta_k0) (rad)
%                       LOS angle rate (thetadt_k0) (rad/sec)
%                       Kalman filter's previous time step corrected
%                       theta covariance (Ptheta_k0)
%                       Kalman filter's current time step estimate of
%                       range (rng_k1) (m)
%                       Noisy theta measurement (nz_theta) (rad)
%                       Estimated magnitude of combined deterministic
%                       missile acceleration perpendicular to the LOS
%                       (mac_perpL_k1).
%    Outputs:           Kalman filter's current time step estimate of:
%                       LOS angle (theta_k1) (rad)
%                       LOS angle rate (thetadt_k1) (rad/sec)
%                       Kalman filter's current time step corrected
%                       theta covariance (Ptheta_k1)
%    Process:
%    Assumptions:
%    Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%------ define globals ------
global DELTA SIG_THETA Q2_PN_THETA GRAV TRN_G
%------ define constants ------
% Transition matrix
F = [1, DELTA;
     0,     1];

% Covariance matrix for uncorrelated bearing measurements
R = (SIG_THETA^2);

% Measurement matrix
H =[1, 0];

% Plant noise gain matrix
G = eye(2);

%------ define input vector ------
```

135

```matlab
% Set corrected theta state values from previous discrete time step
xc = [    theta_k0;
        thetadt_k0];

Pc = Ptheta_k0;

%------ initialize variables ------
% Plant noise covariance multiplier
q2 = Q2_PN_THETA*3*DELTA*((TRN_G*GRAV)^2)/(4*(rng_k1^2));

% Plant noise covariance
Q = q2*[(DELTA^3)/3, (DELTA^2)/2;
        (DELTA^2)/2,      DELTA];

%------ functions ------
%% Prediction phase
% Predicted theta state
u = [    atan2(mac_perpL_k1*GRAV*(DELTA^2), 2*rng_k1);
                    4*mac_perpL_k1*GRAV*rng_k1*DELTA / ...
            (4*(rng_k1^2) + (mac_perpL_k1*GRAV)^2*(DELTA^4))];
                % Position and velocity portions of the missile's
                % deterministic acceleration inputs only.
xp = F*xc + G*u;    % Predicted theta state

% Predicted range state estimation covariance
Pp = F*Pc*F' + Q;

%% Correction Phase
% Calculate Kalman Gain (W)
S = R + H*Pp*H';     % Innovation covariance
W = S\(Pp*H'); % A\b used vice inv(A)*b for speed of computation.

% Calculate the innovations
v = nz_theta - H*xp;

% Calculate corrected state vector (xc)
xc = xp + W*v;

% Calculate corrected covariance (Pc)
Pc = (eye(2)-W*H)*Pp*(eye(2)-W*H)' + W*R*W';

% Corrected theta state values for current discrete time step
theta_k1 = xc(1);
thetadt_k1 = xc(2);

Ptheta_k1 = Pc;

end
```

LIST OF REFERENCES

[1]   R. Westrum, *Sidewinder: Creative Missile Development at China Lake*. Annapolis, MD: Naval Institute Press, 1999.

[2]   K. R. Mayer. (1993). "The development of the advanced medium-range air-to-air missile: A case study of risk and reward in weapon system acquisition" (N-3620-AF). RAND, Santa Monica, CA. [Online]. Available: http://www.rand.org/content/dam/rand/pubs/notes/2009/N3620.pdf

[3]   U.S. Congressional Budget Office. (1986). "The advanced medium-range air-to-air missile (AMRAAM): Current plans and alternatives," Congr. Budget Office, Washington, DC, Staff Working Paper. [Online]. Available: http://www.dtic.mil/get-tr-doc/pdf?AD=ADA529898

[4]   *Jane's HIS*. (July 2013). "AIM-54 Phoenix record." [Online]. Available: https://janes.ihs.com.libproxy.nps.edu/CustomPages/Janes/DisplayPage.aspx?ShowProductLink=true&ShowProductLink=true&DocType=Reference&ItemId=+++1307241&Pubabbrev=JALW

[5]   *Jane's HIS*. (November 2013). AIM-120 AMRAAM record. [Online]. Available: https://janes.ihs.com.libproxy.nps.edu/CustomPages/Janes/DisplayPage.aspx?DocType=Reference&ItemId=+++1307244&Pubabbrev=JALW

[6]   U.S. General Accounting Office. (1991). "Missile procurement: AMRAAM's reliability is improving, but production challenges remain" (report AD-A242 140). U. S. General Accounting Office, Washington, DC. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a242140.pdf

[7]   Raytheon Corp. (2013, August). "US Army, US Air Force intercept cruise missile for first time with JLENS-guided AMRAAM" [news release]. [Online]. Available: http://raytheon.mediaroom.com/index.php?s=43&item=2392

[8]   U.S. Air Force. (2013, April). "RDT&E budget item justification." [Online]. Available: http://www.dtic.mil/descriptivesum/Y2014/AirForce/stamped/0604330F_4_PB_2014.pdf

[9]   P. Zarchan, Ed., *Tactical and Strategic Missile Guidance*. 3rd ed., Reston, VA: Amer. Inst. of Aeronautics and Astronautics, Inc., 1997.

[10]   D. Pehr, "A study into advanced guidance laws using computational methods," M.S. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2011.

[11]   R. Broadston, "A method of increasing the kinematic boundary of air-to-air missiles using an optimal control approach," Elect. Eng. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2000.

[12]   C. Li et al., "Application of 2D differential geometric guidance to tactical missile interception," presented at the IEEE Aerospace Conference, Big Sky, MT, 2006.

[13]   U. S. Shukla and P. R. Mahapatra, "The proportional navigation dilemma—pure or true?," *IEEE Trans. on Aerospace and Electron. Syst.*, vol. 26, no. 2, pp. 382–392, March 1990.

[14]   R. G. Hutchins, "Navigation, missile, and avionics systems," class notes [EC 4330], Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, January 2005.

[15]   Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. New York, NY: John Wiley & Sons, Inc., 2001.

[16]   R. G. Hutchins, "Optimal estimation, Kalman filters and target tracking," EC 3310 course notes, Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, March 1997.

[17]   B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*, 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc., 2003.

INITIAL DISTRIBUTION LIST

1.   Defense Technical Information Center
        Ft. Belvoir, Virginia

2.   Dudley Knox Library
        Naval Postgraduate School
        Monterey, California